Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное учреждение высшего образования
ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
СИСТЕМ УПРАВЛЕНИЯ И РАДИОЭЛЕКТРОНИКИ (ТУСУР)
Кафедра компьютерных систем в управлении и проектировании (КСУП)

СОЗДАНИЕ ПРОСТОГО ПРИЛОЖЕНИЯ ДЛЯ РАБОТЫ С БАЗАМИ ДАННЫХ В MICROSOFT VISUAL STUDIO

Отчет по лабораторной работе по дисциплине «Основы разработки баз данных»

Студент гр. 573-3:

_____ Р.В. Слиньков
дата

_____
подпись

Руководитель:

Преподаватель:

_____ _____ Р. О. Остапенко
оценка подпись

_____
дата

Томск, 2025

Введение

В современном мире базы данных играют ключевую роль в системах хранения и обработки информации. Они используются в самых разных областях — от небольших веб-приложений до крупных корпоративных систем. Эффективная организация данных позволяет оптимизировать работу программных продуктов, обеспечить целостность и безопасность информации, а также упростить её поиск и анализ.

Цель данной лабораторной работы:

1) Изучение основных компонент для работы с данными в базах данных в Microsoft Visual Studio;

2) Создание простого приложения в Microsoft Visual Studio для работы с базой данных для своей предметной области (на примере базы данных для кафе).

Задачи:

Реализовать программу согласно требованиям:

1. Создать главную форму проекта с наличием главного меню, контекстного меню, панели инструментов;
2. Настроить, чтобы главная форма сохраняла свое положение при повторном запуске программы; БД должна состоять минимум из 3-4 связанных таблиц (на каждого студента);
3. Провести подключение к БД в приложении;
4. Сформировать конфигурационный файл для изменения пути к БД в приложении;
5. Провести визуальное оформление данных БД в приложении. Обязательно:
    a. Наличие нескольких форм в проекте;
    b. Использование различных элементов управления Label, TextBox, ComboBox (и/или ListBox), CheckBox, PictureBox, DataGridView, BindingNavigator и др.

6. Провести модификацию настроек DataGridView (типы колонок обязательно);

7. Настройте для удобства BindingNavigator (всплывающие подсказки обязательно;

8. Предоставить работу с графическим полем (в том числе загрузку фотографии из файла);

9. Реализация шаблона проектирования «Одиночка» (англ. Singleton);

10. Протестируйте работу полученного приложения.

1 ОСНОВНАЯ ЧАСТЬ

1.1 Описание архитектуры для создания приложения и ход выполнения работы

Windows Forms — это технология для создания настольных приложений на платформе .NET, которая предоставляет мощный инструментарий для разработки пользовательского интерфейса. Она позволяет использовать широкий набор элементов управления, таких как Label для отображения текста, TextBox для ввода данных, ComboBox для выбора из списка, CheckBox для выбора опций, PictureBox для работы с изображениями, DataGridView для отображения данных в табличном виде и BindingNavigator для навигации по записям. Эти элементы обеспечивают гибкость и удобство при создании интерактивных приложений.

В рамках данной лабораторной работы была реализована архитектура приложения, основанная на принципах разделения ответственности и модульности. Для взаимодействия с базой данных использовался **паттерн репозитория**, который представляет собой прослойку между бизнес-логикой приложения и уровнем доступа к данным. Этот паттерн реализован через интерфейсы (IWarehouseRepository, ISupplierRepository, ISupplyRepository, IStorageZoneRepository, IEmployeeRepository) и их конкретные реализации в проекте Infrastructure. Каждый репозиторий предоставляет методы для выполнения операций CRUD (Create, Read, Update, Delete) над соответствующей сущностью, например, GetAll, Add, Update, Delete.

Почему была выбрана такая архитектура?

Архитектура приложения была спроектирована с использованием паттерна репозитория для достижения нескольких целей:

- **Разделение ответственности**: Бизнес-логика (например, валидация данных в формах) отделена от логики доступа к данным (работа с базой данных). Это делает код более читаемым и поддерживаемым.

• **Изоляция слоя данных**: Репозитории скрывают детали работы с базой данных (например, SQL-запросы, подключение к базе) от остальной части приложения. Это позволяет легко заменить технологию базы данных (например, с SQLite на SQL Server) без изменения кода форм.

Для управления зависимостями в приложении применён паттерн Singleton, реализованный в классе AppContext. Этот класс предоставляет единую точку доступа к репозиториям через статическое свойство Instance. Например, в формах репозитории получаются следующим образом:

```
_employeeRepository=Application.AppContext.Instance.Employee
Repository;
```

Плюсы выбранной архитектуры:

• **Модульность**: Приложение разделено на проекты (Domain, Infrastructure, UI, Application), что упрощает масштабирование и поддержку кода. Например, проект Domain содержит модели и интерфейсы, Infrastructure — реализацию репозиториев, а UI — пользовательский интерфейс.

• **Гибкость**: Благодаря интерфейсам и паттерну репозитория можно легко заменить слой данных или добавить новую функциональность (например, кэширование данных).

• **Поддерживаемость**: Код организован по принципу единственной ответственности (Single Responsibility Principle), что упрощает его понимание и модификацию.

Таким образом, использование паттернов репозитория и Singleton, а также разделение приложения на проекты, позволило создать гибкую, модульную и тестируемую систему, которая соответствует принципам чистой архитектуры и упрощает дальнейшую разработку.

Ход выполнения заданий

1. **Создание главной формы:**

Была создана главная форма MainForm с главным меню (MenuStrip), контекстным меню (ContextMenuStrip) и панелью инструментов (ToolStrip). Главное меню содержит пункты для открытия форм WarehouseForm,

SupplierForm, SupplyForm, StorageZoneForm, EmployeeForm и AboutForm. Панель инструментов предоставляет быстрый доступ к этим формам через кнопки.

2. **Сохранение положения главной формы:**

Реализовано сохранение положения и размера MainForm при закрытии приложения с использованием Properties.Settings. Данные загружаются при следующем запуске.

3. **Подключение к базе данных:**

Подключение к базе данных настроено через строку подключения в файле конфигурации app.config. Использован паттерн репозитория для взаимодействия с базой данных через интерфейсы, такие как IWarehouseRepository, ISupplierRepository, IEmployeeRepository и др.

4. **Создание конфигурационного файла:**

В файл app.config добавлена строка подключения, которая позволяет изменять путь к базе данных без изменения кода приложения.

5. **Визуальное оформление данных:**

Разработаны формы WarehouseForm, SupplierForm, SupplyForm, StorageZoneForm, EmployeeForm и AboutForm. Использованы элементы управления Label, TextBox, ComboBox, PictureBox, DataGridView, BindingNavigator. В EmployeeForm добавлена возможность загрузки фотографии сотрудника через PictureBox.

6. **Модификация настроек DataGridView:**

В форме SupplyForm настроены типы колонок DataGridView: для колонок ProductId и SupplierId использован тип DataGridViewComboBoxColumn для отображения названий продуктов и поставщиков вместо их идентификаторов.

7. **Настройка BindingNavigator:**

Во всех формах добавлен BindingNavigator с всплывающими подсказками для кнопок навигации, добавления и удаления записей. В EmployeeForm исправлена проблема с удалением записей через

BindingNavigator, добавлен обработчик события DeleteItem для синхронизации с базой данных.

8. **Работа с графическим полем:**

В EmployeeForm реализована загрузка фотографии сотрудника через OpenFileDialog и отображение её в PictureBox. Фотография сохраняется в базе данных в виде массива байтов.

9. **Реализация паттерна Singleton:**

Паттерн Singleton применён в классе AppContext для предоставления единого доступа к репозиториям. Это обеспечивает централизованное управление зависимостями.

1.2 Результаты выполнения

В результате выполнения лабораторной работы было создано полнофункциональное Windows Forms приложение "Система складского учёта", соответствующее требованиям технического задания. Приложение позволяет управлять данными о складах, поставщиках, поставках, зонах хранения и сотрудниках. Реализованы следующие функции:

- Главная форма MainForm предоставляет удобный интерфейс для навигации между формами через главное меню и панель инструментов. Положение формы сохраняется между запусками. (рисунок 1.1)
- Формы для работы с сущностями (WarehouseForm (рисунок 1.2), SupplierForm (рисунок 1.3), SupplyForm (рисунок 1.4), StorageZoneForm (рисунок 1.5), EmployeeForm (рисунок 1.6), ProductAccountingForm (рисунок 1.7), ProductForm (рисунок 1.8)) позволяют просматривать, добавлять, редактировать и удалять записи. Данные отображаются в DataGridView с настроенными типами колонок.
- Форма AboutForm содержит информацию о приложении, включая название, версию, данные разработчика и ссылку на GitHub-репозиторий. Также отображается логотип приложения (рисунок 1.9).
- Реализована загрузка и отображение фотографий сотрудников в EmployeeForm с использованием PictureBox и OpenFileDialog (рисунок 1.10 и рисунок 1.11).
- Подключение к базе данных настроено через App.config, что позволяет легко изменять путь к базе данных (ПРИЛОЖЕНИЕ А - Листинг).
- Паттерн Singleton в AppContext обеспечивает централизованное управление репозиториями (ПРИЛОЖЕНИЕ А - Листинг).

Рисунок 1.1 – Внешний вид главной формы

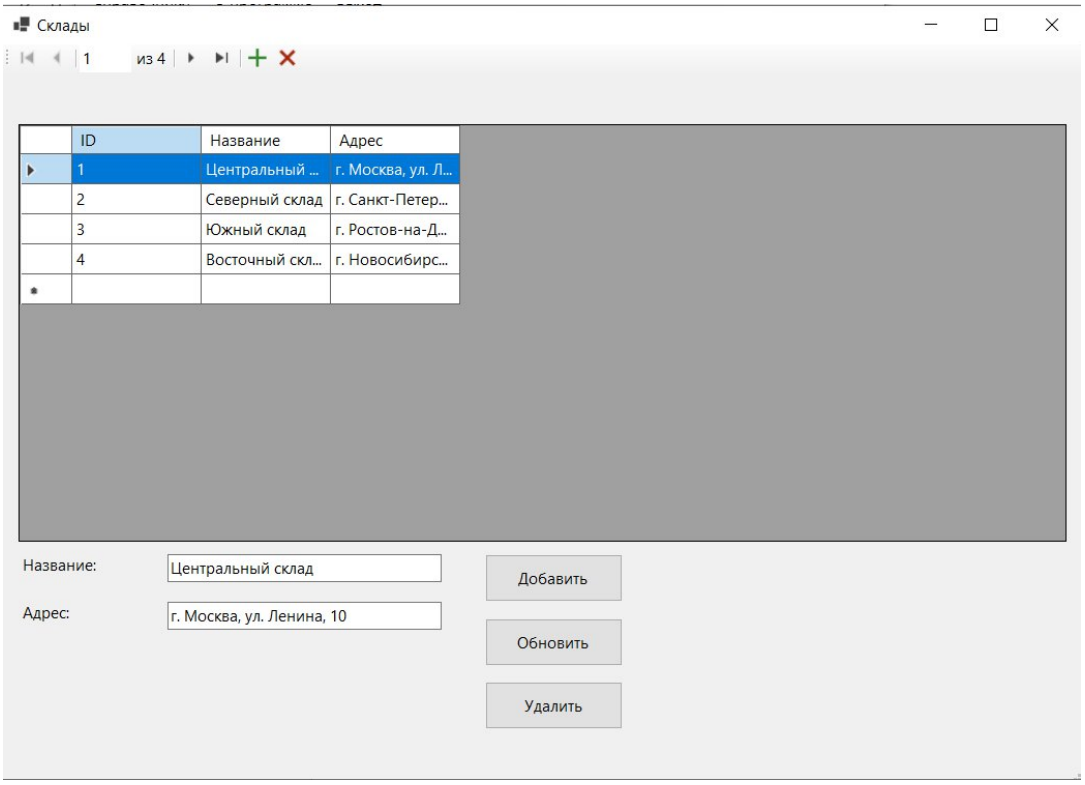

Рисунок 1.2 – Внешний вид WarehouseForm

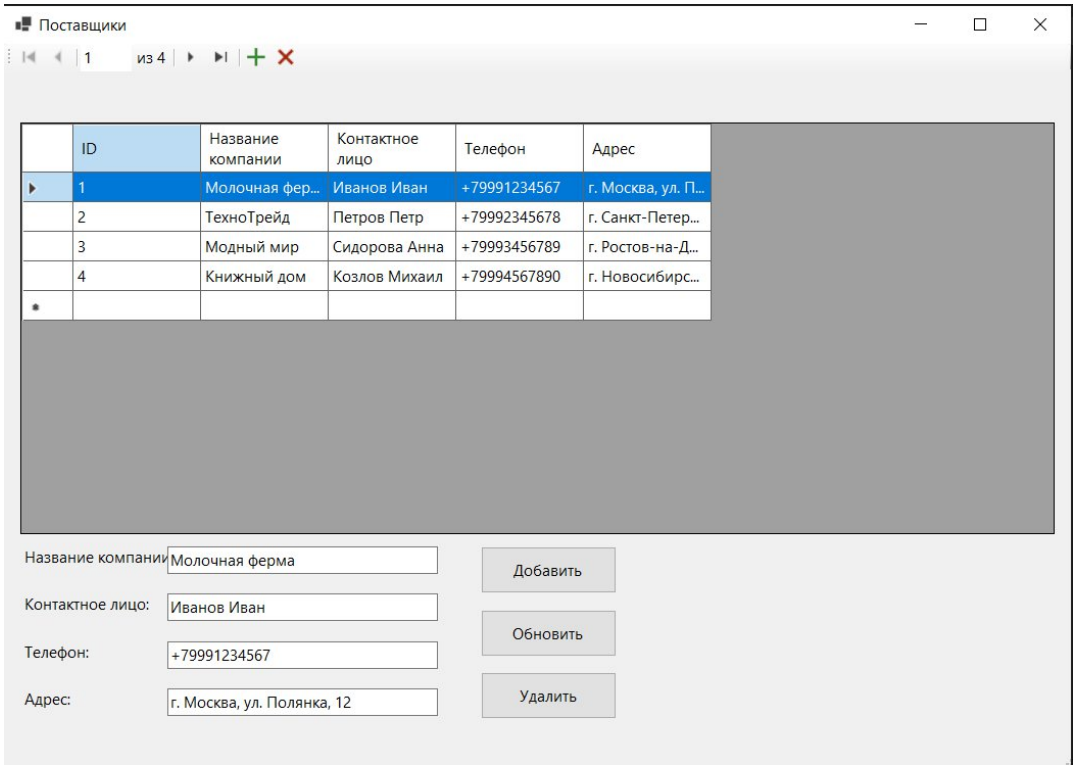

Рисунок 1.3 – Внешний вид SupplierForm

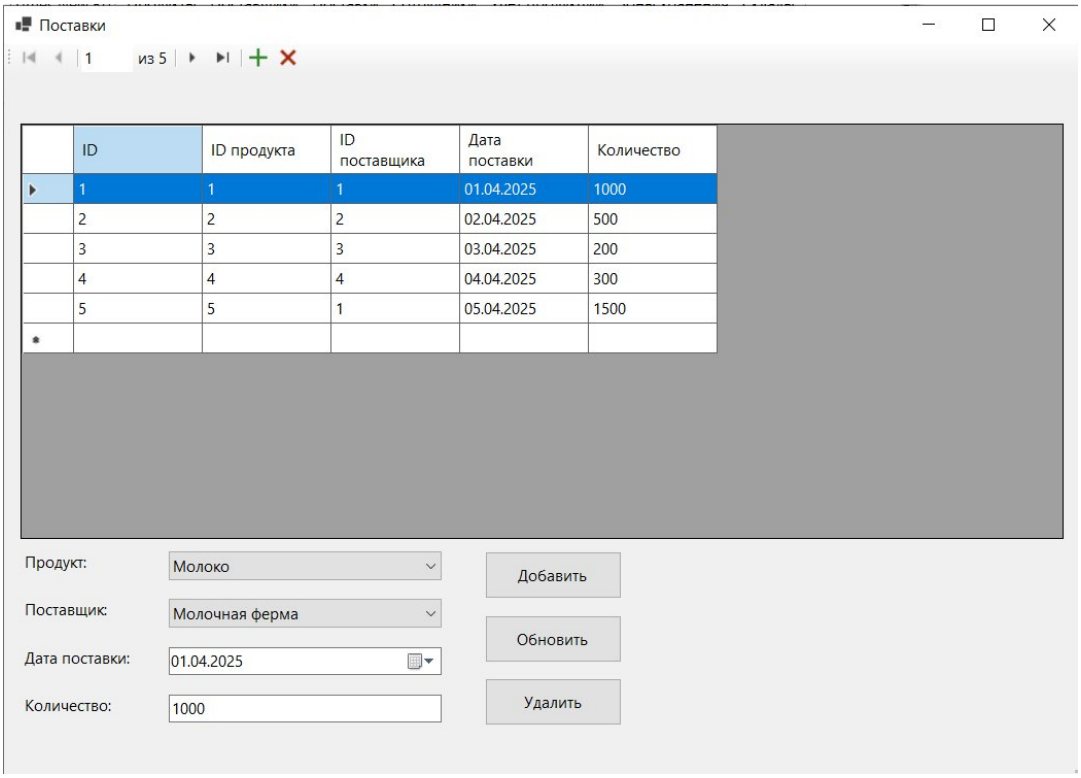

Рисунок 1.4 – Внешний вид SupplyForm

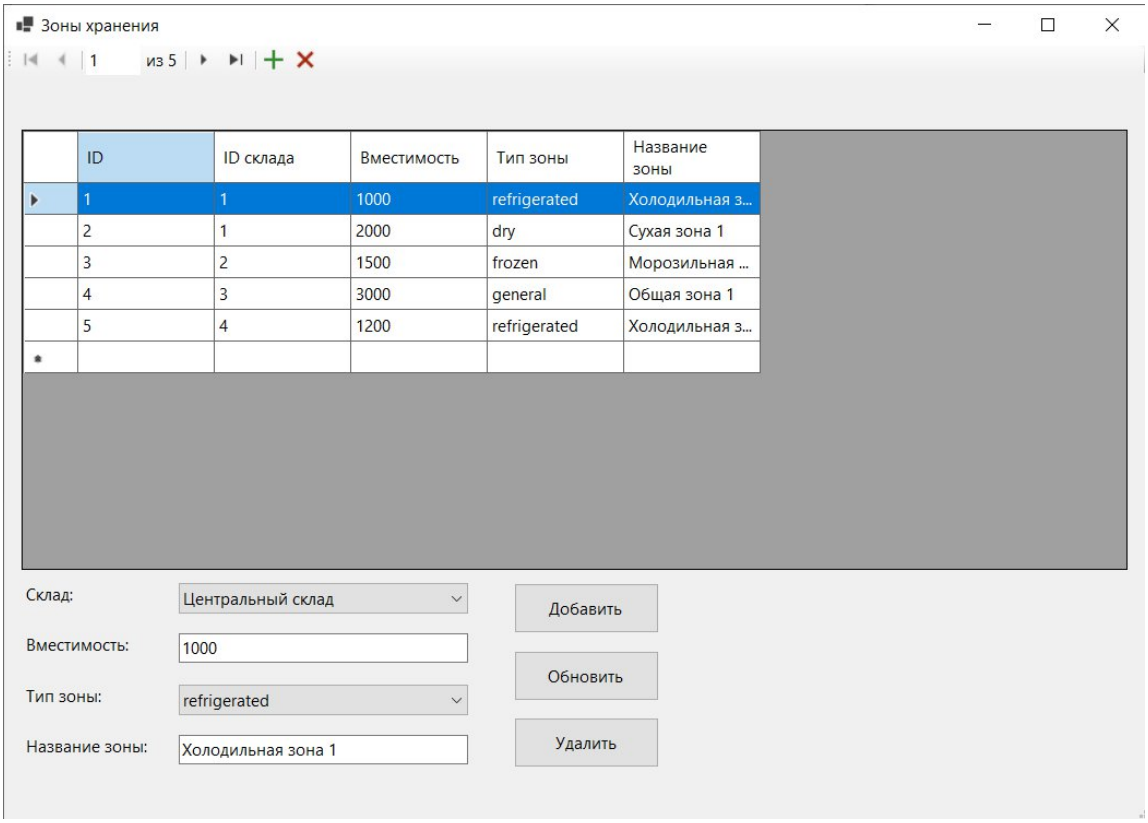

Рисунок 1.5 – Внешний вид StorageZoneForm

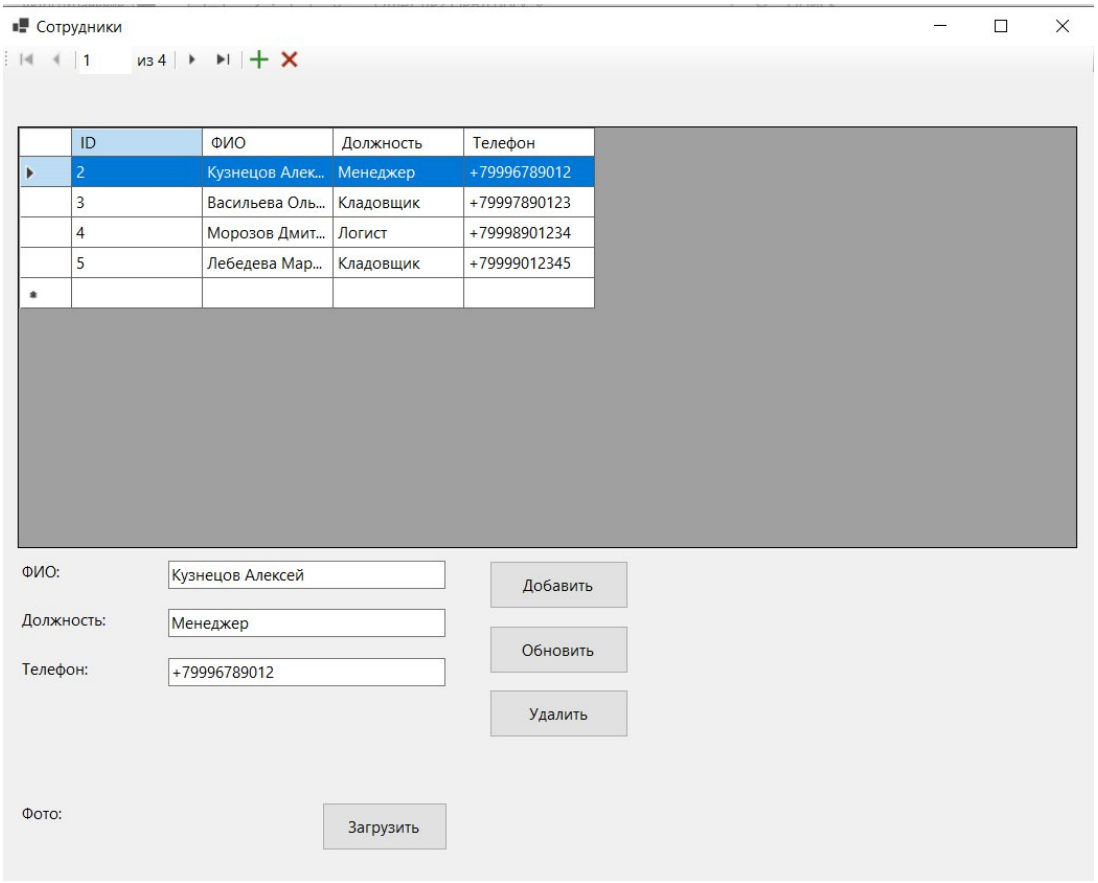

Рисунок 1.6 – Внешний вид EmployeeForm

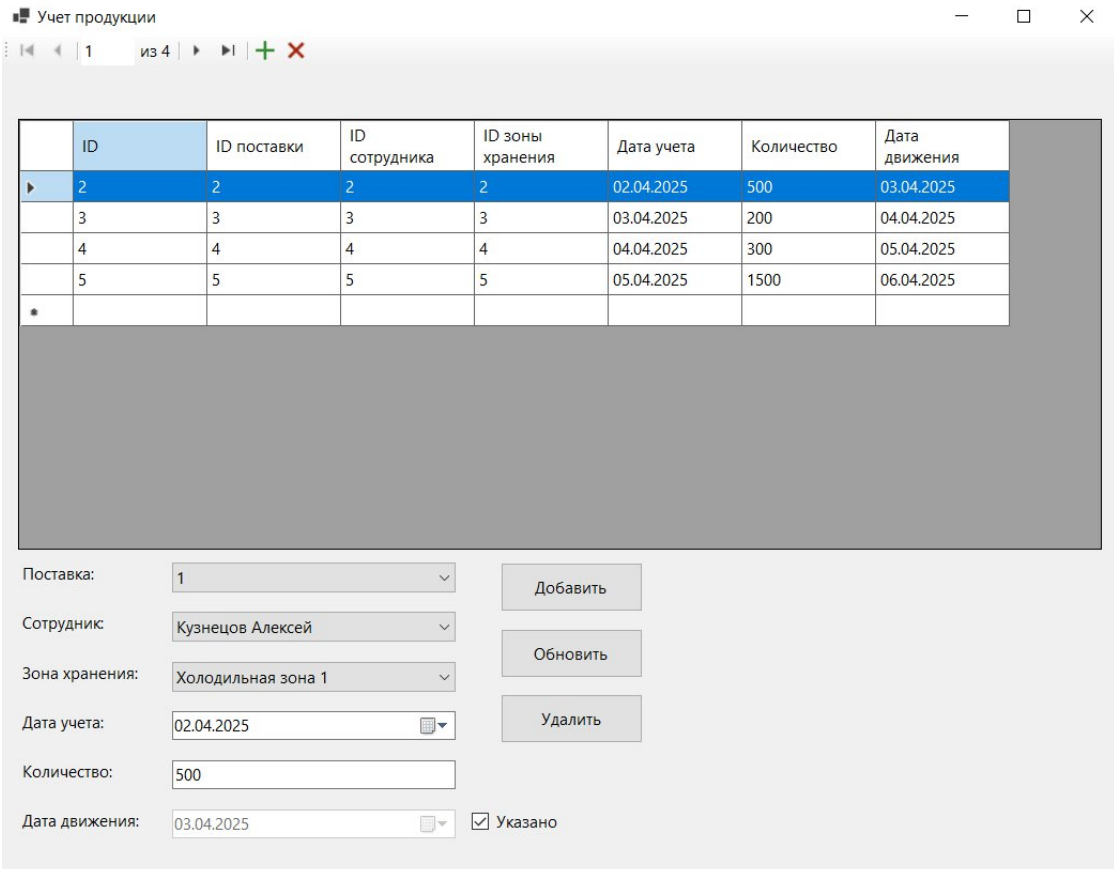

Рисунок 1.7 – Внешний вид ProductAccountingForm

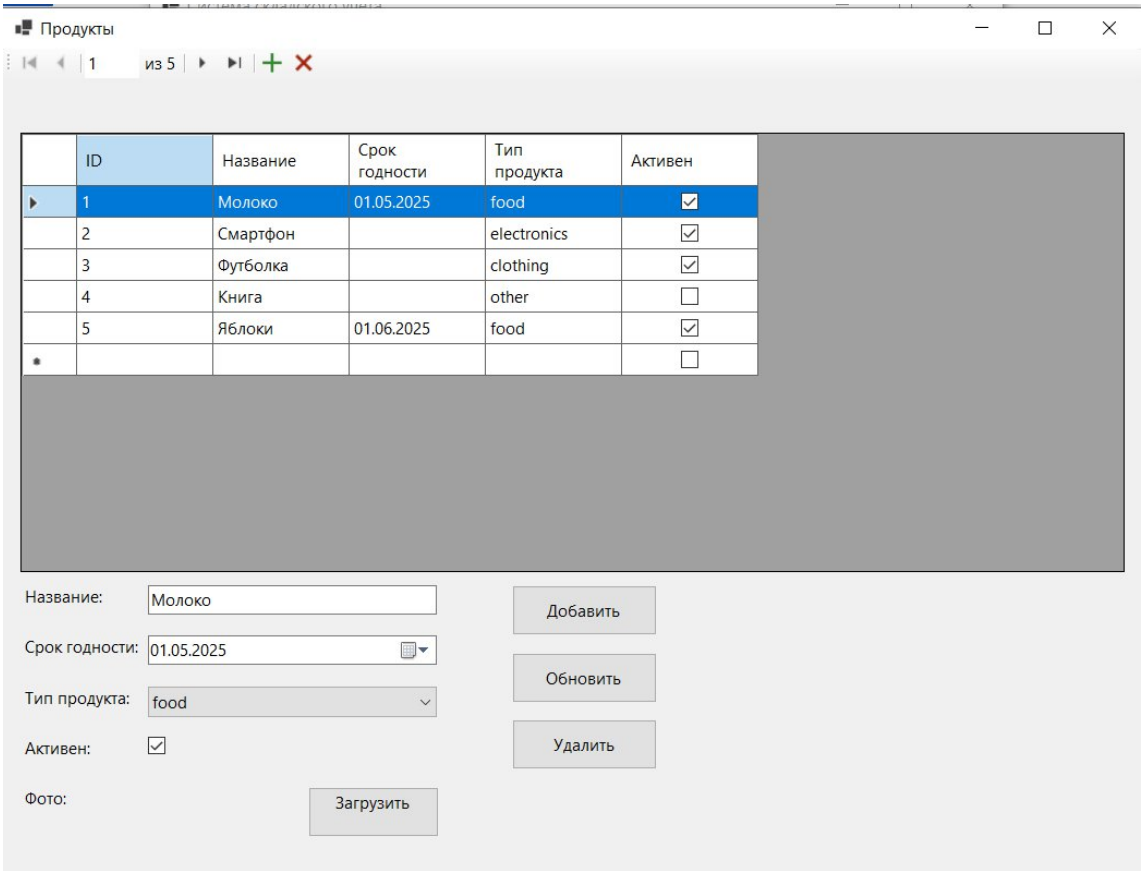

Рисунок 1.8 – Внешний вид ProductForm

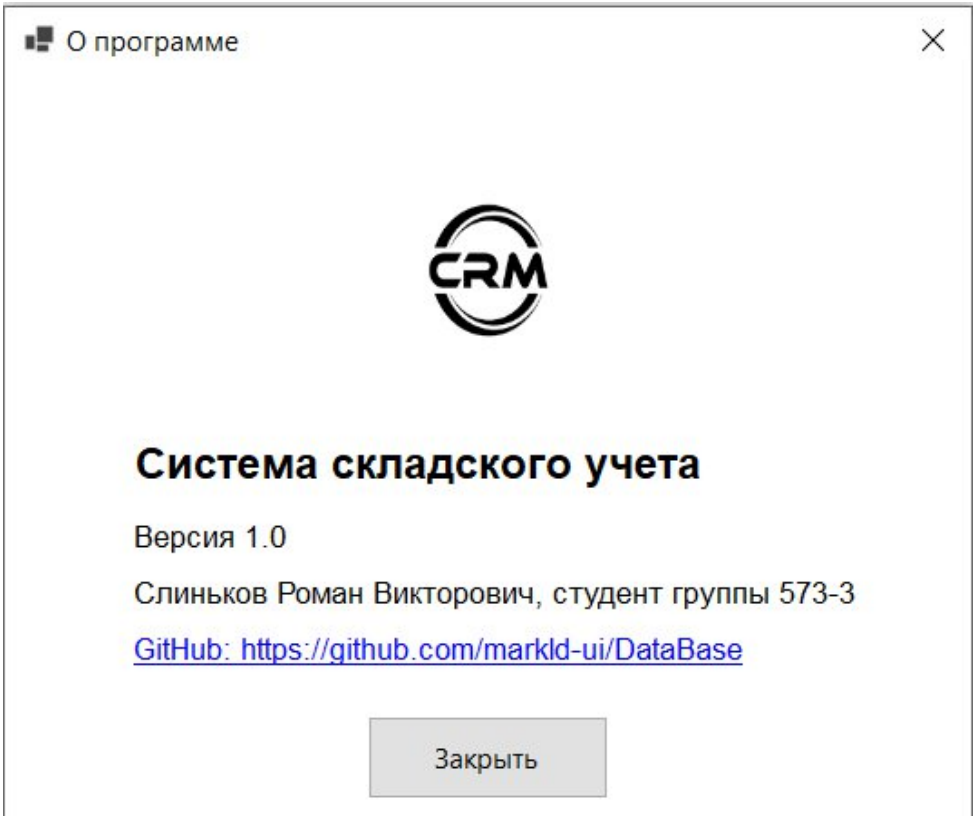

Рисунок 1.9 – Внешний вид AboutForm

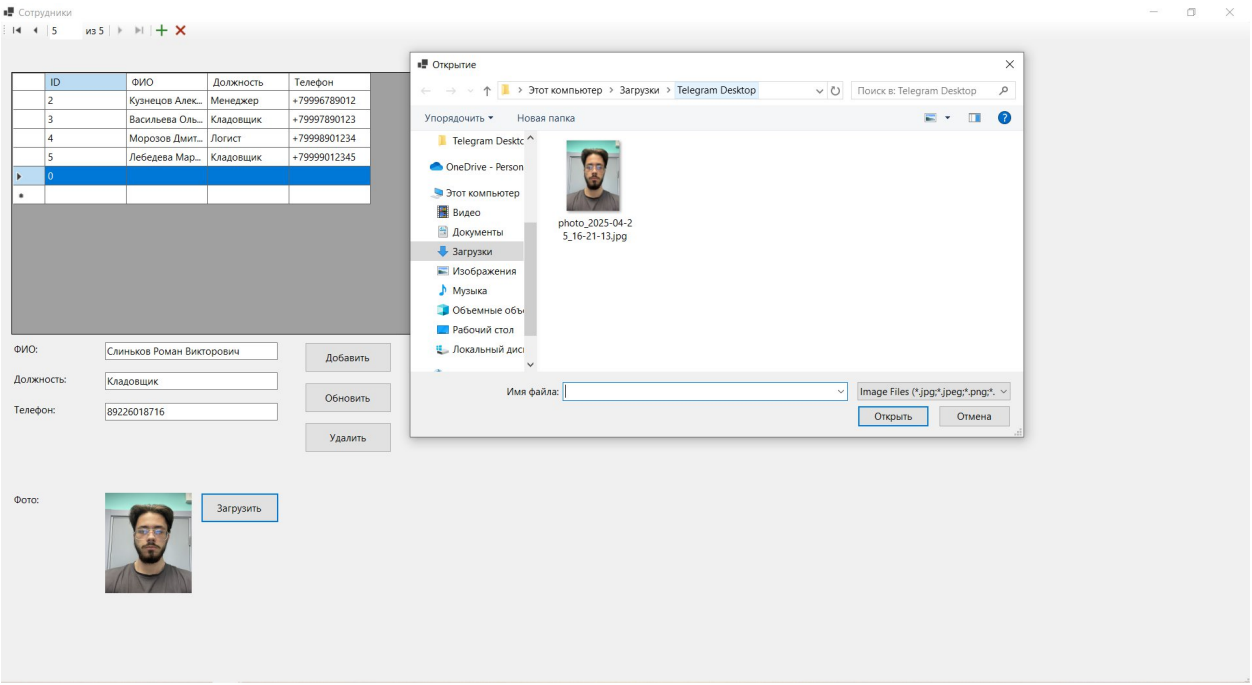

Рисунок 1.10 – Загрузка фотографии в EmployeeForm

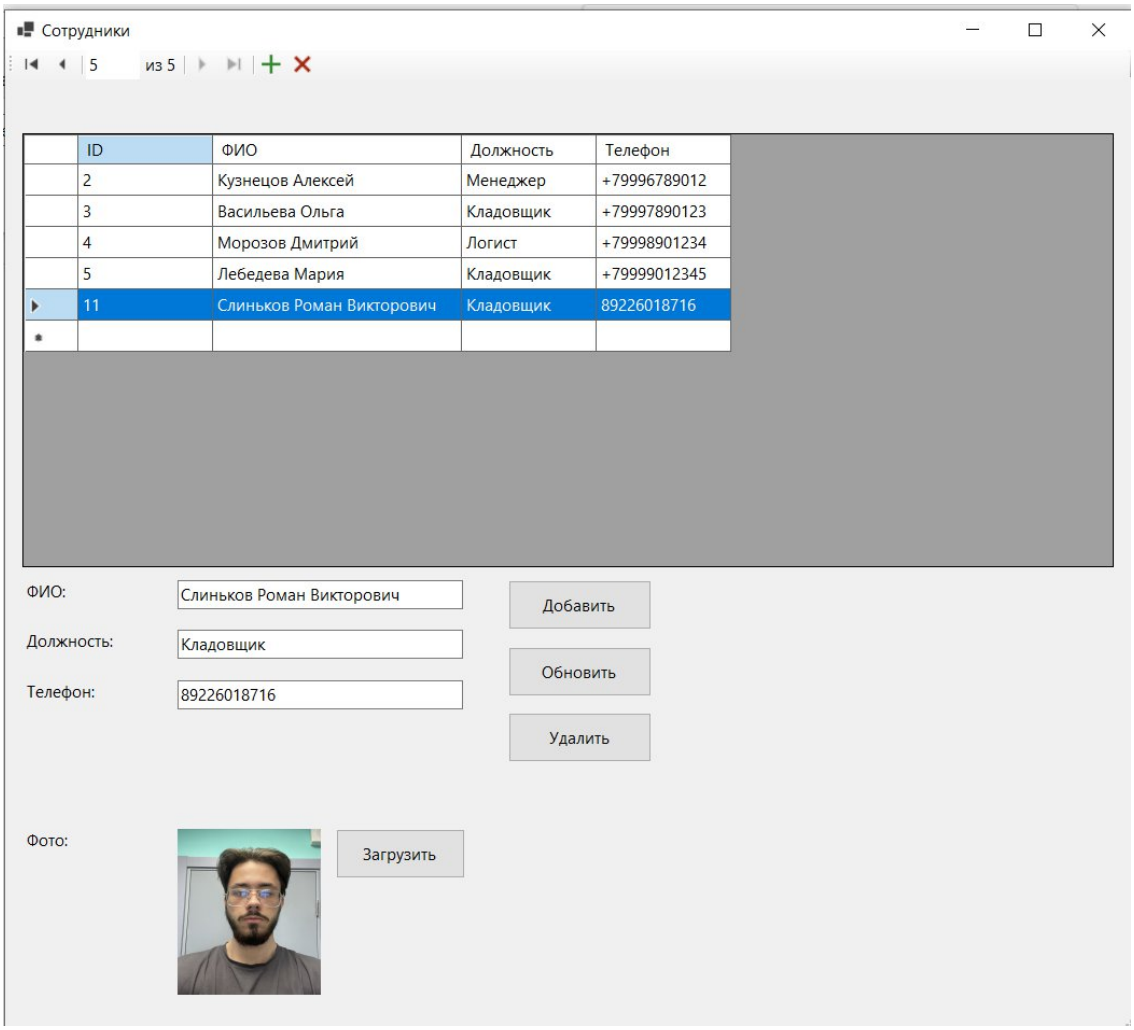

Рисунок 1.11 – Отображение фотографии в EmployeeForm

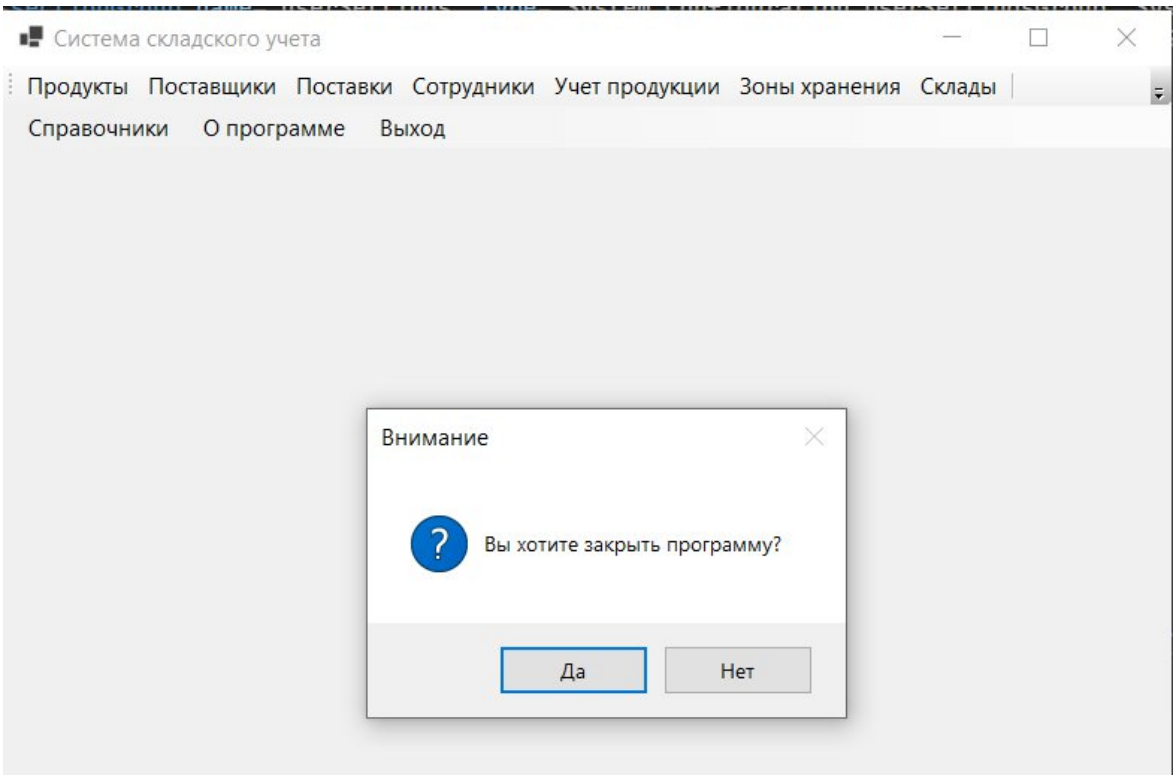

Рисунок 1.12 – Закрытие приложения

Заключение

В ходе выполнения лабораторной работы было разработано Windows Forms приложение "Система складского учёта", которое полностью соответствует требованиям технического задания. Приложение предоставляет удобный пользовательский интерфейс для управления данными о складах, поставщиках, поставках, зонах хранения и сотрудниках, обеспечивая выполнение операций просмотра, добавления, редактирования и удаления записей.

Ключевые результаты работы:

- Реализованы все пункты лабораторной работы, включая создание главной формы с меню и панелью инструментов, сохранение её положения, подключение к базе данных через конфигурационный файл, визуальное оформление данных, настройку DataGridView и BindingNavigator, работу с графическим полем и применение паттерна Singleton.
- Архитектура приложения спроектирована с использованием паттерна репозитория, что обеспечило разделение бизнес-логики и доступа к данным, повысило модульность и тестируемость кода. Паттерн Singleton, реализованный в классе AppContext, позволил централизованно управлять зависимостями.
- Добавлена поддержка загрузки и отображения фотографий сотрудников через PictureBox, что соответствует требованиям пункта 5.2.8.

Приложение демонстрирует высокую степень модульности и гибкости благодаря разделению на проекты (Domain, Infrastructure, UI, Application). Использование интерфейсов и паттернов проектирования упрощает дальнейшую поддержку и расширение функциональности, например, добавление новых сущностей или замену технологии базы данных.

ПРИЛОЖЕНИЕ А

Листинг проекта Application

Листинг файла AppContext.cs:

```
using System;
using Common;
using Infrastructure;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using Domain.Interfaces;
using Infrastructure.Repositories;

namespace Application
{
    public class AppContext
    {
        private static AppContext _instance;
        private static readonly object _lock = new object();
        private readonly DatabaseConnection _dbConnection;
        private readonly IProductRepository _productRepository;
      private readonly ISupplierRepository _supplierRepository;
        private readonly ISupplyRepository _supplyRepository;
      private readonly IEmployeeRepository _employeeRepository;
      private readonly IWarehouseRepository _warehouseRepository;
        private readonly IStorageZoneRepository
_storageZoneRepository;
        private readonly IProductAccountingRepository
_productAccountingRepository;

        private AppContext()
        {
```

```
            string connectionString = Common.ConfigurationManager.GetConnectionString("WarehouseConnection");
            _dbConnection = new DatabaseConnection(connectionString);
            _productRepository = new ProductRepository(_dbConnection);
            _supplierRepository = new SupplierRepository(_dbConnection);
         _supplyRepository = new SupplyRepository(_dbConnection);
            _employeeRepository = new EmployeeRepository(_dbConnection);
            _warehouseRepository = new WarehouseRepository(_dbConnection);
            _storageZoneRepository = new StorageZoneRepository(_dbConnection);
            _productAccountingRepository = new ProductAccountingRepository(_dbConnection);
        }

        public static AppContext Instance
        {
            get
            {
                if (_instance == null)
                {
                    lock (_lock)
                    {
                        if (_instance == null)
                        {
                            _instance = new AppContext();
                        }
                    }
                }
                return _instance;
            }
        }
```

```
        public     IProductRepository     ProductRepository     =>
_productRepository;
        public    ISupplierRepository    SupplierRepository    =>
_supplierRepository;
        public     ISupplyRepository     SupplyRepository     =>
_supplyRepository;
        public    IEmployeeRepository    EmployeeRepository    =>
_employeeRepository;
        public    IWarehouseRepository   WarehouseRepository   =>
_warehouseRepository;
        public  IStorageZoneRepository  StorageZoneRepository  =>
_storageZoneRepository;
        public                          IProductAccountingRepository
ProductAccountingRepository => _productAccountingRepository;
        public    DatabaseConnection    DatabaseConnection    =>
_dbConnection;
    }
}
```

Листинг проекта Common

Листинг файла ConfigurationManager.cs:

```
using System;
using System.Configuration;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Common
{
    public static class ConfigurationManager
    {
        public static string GetConnectionString(string name)
        {
```

```
            var connectionString = System.Configuration.ConfigurationManager.ConnectionStrings[name]?.ConnectionString;
            if (string.IsNullOrEmpty(connectionString))
            {
            throw new ConfigurationErrorsException($"Connection string '{name}' not found in App.config.");
            }
            return connectionString;
        }
    }
}
```

Листинг проекта Domain

Листинг файла IEmployeeRepository.cs:

```
using Domain.Models;
using System.Collections.Generic;

namespace Domain.Interfaces
{
    public interface IEmployeeRepository
    {
        List<Employee> GetAll();
        Employee GetById(int id);
        void Add(Employee employee);
        void Update(Employee employee);
        void Delete(int id);
    }
}
```

Листинг файла IProductAccountingRepository.cs:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
```

```
using System.Threading.Tasks;
using Domain.Models;

namespace Domain.Interfaces
{
    public interface IProductAccountingRepository
    {
        List<ProductAccounting> GetAll();
        ProductAccounting GetById(int id);
        void Add(ProductAccounting productAccounting);
        void Update(ProductAccounting productAccounting);
        void Delete(int id);
    }
}
```

Листинг файла IProductRepository.cs:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using Domain.Models;

namespace Domain.Interfaces
{
    public interface IProductRepository
    {
        List<Product> GetAll();
        Product GetById(int id);
        void Add(Product product);
        void Update(Product product);
        void Delete(int id);
    }
}
```

Листинг файла IStorageZoneRepository.cs:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using Domain.Models;

namespace Domain.Interfaces
{
    public interface IStorageZoneRepository
    {
        List<StorageZone> GetAll();
        StorageZone GetById(int id);
        void Add(StorageZone storageZone);
        void Update(StorageZone storageZone);
        void Delete(int id);
    }
}
```

Листинг файла ISupplierRepository.cs:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using Domain.Models;

namespace Domain.Interfaces
{
    public interface ISupplierRepository
    {
        List<Supplier> GetAll();
```

```
        Supplier GetById(int id);
        void Add(Supplier supplier);
        void Update(Supplier supplier);
        void Delete(int id);
    }
}
```

Листинг файла ISupplyRepository.cs:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using Domain.Models;

namespace Domain.Interfaces
{
    public interface ISupplyRepository
    {
        List<Supply> GetAll();
        Supply GetById(int id);
        void Add(Supply supply);
        void Update(Supply supply);
        void Delete(int id);
    }
}
```

Листинг файла IWarehouseRepository.cs:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
```

```
using System.Threading.Tasks;
using Domain.Models;

namespace Domain.Interfaces
{
    public interface IWarehouseRepository
    {
        List<Warehouse> GetAll();
        Warehouse GetById(int id);
        void Add(Warehouse warehouse);
        void Update(Warehouse warehouse);
        void Delete(int id);
    }
}
```

Листинг файла Employee.cs:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Domain.Models
{
    public class Employee
    {
        public int EmployeeId { get; set; }
        public byte[] Photo { get; set; }
        public string FullName { get; set; }
        public string Position { get; set; }
        public string Phone { get; set; }
    }
}
```

Листинг файла Product.cs:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Domain.Models
{
    public class Product
    {
        public int ProductId { get; set; }
        public string Name { get; set; }
        public DateTime? ExpiryDate { get; set; }
        public ProductType ProductType { get; set; }
        public bool IsActive { get; set; }
        public byte[] Photo { get; set; }
    }
}
```

Листинг файла ProductAccounting.cs:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Domain.Models
{
    public class ProductAccounting
    {
        public int ProductAccId { get; set; }
        public int SupplyId { get; set; }
        public int EmployeeId { get; set; }
        public int StorageId { get; set; }
```

```
        public DateTime AccountingDate { get; set; }
        public int Quantity { get; set; }
        public DateTime? LastMovementDate { get; set; }
    }
}
```

Листинг файла ProductType.cs:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Domain.Models
{
    public enum ProductType
    {
        food,
        electronics,
        clothing,
        other
    }
}
```

Листинг файла StorageZone.cs:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Domain.Models
{
    public class StorageZone
```

```
    {
        public int StorageId { get; set; }
        public int WarehouseId { get; set; }
        public int Capacity { get; set; }
        public ZoneType ZoneType { get; set; }
        public string ZoneName { get; set; }
    }
}
```

Листинг файла Supplier.cs:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Domain.Models
{
    public class Supplier
    {
        public int SupplierId { get; set; }
        public string CompanyName { get; set; }
        public string ContactPerson { get; set; }
        public string Phone { get; set; }
        public string Address { get; set; }
    }
}
```

Листинг файла Supply.cs:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
```

```
namespace Domain.Models
{
    public class Supply
    {
        public int SupplyId { get; set; }
        public int ProductId { get; set; }
        public int SupplierId { get; set; }
        public DateTime SupplyDate { get; set; }
        public int Quantity { get; set; }
    }
}
```

Листинг файла Warehouse.cs:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Domain.Models
{
    public class Warehouse
    {
        public int WarehouseId { get; set; }
        public string Name { get; set; }
        public string Address { get; set; }
    }
}
```

Листинг файла ZoneType.cs:

```
using System;
using System.Collections.Generic;
using System.Linq;
```

```
using System.Text;
using System.Threading.Tasks;

namespace Domain.Models
{
    public enum ZoneType
    {
        refrigerated,
        dry,
        frozen,
        general
    }
}
```

Листинг проекта Infrastructure

Листинг файла DatabaseConnection.cs:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using Npgsql;

namespace Infrastructure
{
    public class DatabaseConnection
    {
        private readonly string _connectionString;

        public DatabaseConnection(string connectionString)
        {
            _connectionString = connectionString ?? throw new
ArgumentNullException(nameof(connectionString));
        }
```

```
        public NpgsqlConnection GetConnection()
        {
            return new NpgsqlConnection(_connectionString);
        }

        public bool TestConnection()
        {
            try
            {
                using (var connection = GetConnection())
                {
                    connection.Open();
                    return true;
                }
            }
            catch (Exception ex)
            {
                Console.WriteLine($"Ошибка подключения:
{ex.Message}");
                return false;
            }
        }
    }
}
```

Листинг файла EmployeeRepository.cs:

```
using System;
using Npgsql;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using Domain.Interfaces;
```

```
using Domain.Models;

namespace Infrastructure.Repositories
{
    public class EmployeeRepository : IEmployeeRepository
    {
        private readonly DatabaseConnection _dbConnection;

      public EmployeeRepository(DatabaseConnection dbConnection)
        {
            _dbConnection = dbConnection ?? throw new ArgumentNullException(nameof(dbConnection));
        }

        public List<Employee> GetAll()
        {
            var employees = new List<Employee>();
            using (var conn = _dbConnection.GetConnection())
            {
                conn.Open();
                using (var cmd = new NpgsqlCommand("SELECT employee_id, photo, full_name, position, phone FROM employee", conn))
                using (var reader = cmd.ExecuteReader())
                {
                    while (reader.Read())
                    {
                        employees.Add(new Employee
                        {
                            EmployeeId = reader.GetInt32(0),
                            Photo = reader.IsDBNull(1) ? null : (byte[])reader.GetValue(1),
                            FullName = reader.GetString(2),
                            Position = reader.GetString(3),
```

```
                            Phone = reader.IsDBNull(4) ? null :
reader.GetString(4)
                        });
                    }
                }
            }
            return employees;
        }

        public Employee GetById(int id)
        {
            using (var conn = _dbConnection.GetConnection())
            {
                conn.Open();
                using (var cmd = new NpgsqlCommand("SELECT
employee_id, photo, full_name, position, phone FROM employee WHERE
employee_id = @id", conn))
                {
                    cmd.Parameters.AddWithValue("id", id);
                    using (var reader = cmd.ExecuteReader())
                    {
                        if (reader.Read())
                        {
                            return new Employee
                            {
                                EmployeeId = reader.GetInt32(0),
                        Photo     =     reader.IsDBNull(1)     ?
null : (byte[])reader.GetValue(1),
                                FullName = reader.GetString(2),
                                Position = reader.GetString(3),
                        Phone     =     reader.IsDBNull(4)     ?
null : reader.GetString(4)
                            };
                        }
                        return null;
```

```
                }
            }
        }
    }

    public void Add(Employee employee)
    {
        using (var conn = _dbConnection.GetConnection())
        {
            conn.Open();
            using (var cmd = new NpgsqlCommand(
           "INSERT INTO employee (photo, full_name, position,
phone) VALUES (@photo, @full_name, @position, @phone) RETURNING
employee_id", conn))
            {
                cmd.Parameters.AddWithValue("photo",
(object)employee.Photo ?? DBNull.Value);
                cmd.Parameters.AddWithValue("full_name",
employee.FullName);
                cmd.Parameters.AddWithValue("position",
employee.Position);
                cmd.Parameters.AddWithValue("phone",
(object)employee.Phone ?? DBNull.Value);
           employee.EmployeeId = (int)cmd.ExecuteScalar();
            }
        }
    }

    public void Update(Employee employee)
    {
        using (var conn = _dbConnection.GetConnection())
        {
            conn.Open();
            using (var cmd = new NpgsqlCommand(
```

```
                "UPDATE employee SET photo = @photo, full_name
= @full_name, position = @position, phone = @phone WHERE
employee_id = @id", conn))
                {
                    cmd.Parameters.AddWithValue("id",
employee.EmployeeId);
                    cmd.Parameters.AddWithValue("photo",
(object)employee.Photo ?? DBNull.Value);
                    cmd.Parameters.AddWithValue("full_name",
employee.FullName);
                    cmd.Parameters.AddWithValue("position",
employee.Position);
                    cmd.Parameters.AddWithValue("phone",
(object)employee.Phone ?? DBNull.Value);
                    cmd.ExecuteNonQuery();
                }
            }
        }

        public void Delete(int id)
        {
            using (var conn = _dbConnection.GetConnection())
            {
                conn.Open();
                using (var cmd = new NpgsqlCommand("DELETE FROM
employee WHERE employee_id = @id", conn))
                {
                    cmd.Parameters.AddWithValue("id", id);
                    cmd.ExecuteNonQuery();
                }
            }
        }
    }
}
```

Листинг файла ProductAccountingRepository.cs:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using Domain.Interfaces;
using Domain.Models;
using Npgsql;

namespace Infrastructure.Repositories
{
    public class ProductAccountingRepository : IProductAccountingRepository
    {
        private readonly DatabaseConnection _dbConnection;

        public ProductAccountingRepository(DatabaseConnection dbConnection)
        {
            _dbConnection = dbConnection ?? throw new ArgumentNullException(nameof(dbConnection));
        }

        public List<ProductAccounting> GetAll()
        {
         var productAccountings = new List<ProductAccounting>();
            using (var conn = _dbConnection.GetConnection())
            {
                conn.Open();
                using (var cmd = new NpgsqlCommand("SELECT productAcc_id, supply_id, employee_id, storage_id, accounting_date, quantity, last_movement_date FROM product_accounting", conn))
                using (var reader = cmd.ExecuteReader())
                {
```

```
                    while (reader.Read())
                    {
                    productAccountings.Add(new ProductAccounting
                        {
                            ProductAccId = reader.GetInt32(0),
                            SupplyId = reader.GetInt32(1),
                            EmployeeId = reader.GetInt32(2),
                            StorageId = reader.GetInt32(3),
                        AccountingDate = reader.GetDateTime(4),
                            Quantity = reader.GetInt32(5),
                        LastMovementDate                                =
reader.IsDBNull(6) ? null : reader.GetDateTime(6)
                        });
                    }
                }
            }
            return productAccountings;
        }

        public ProductAccounting GetById(int id)
        {
            using (var conn = _dbConnection.GetConnection())
            {
                conn.Open();
                using (var cmd = new NpgsqlCommand("SELECT
productAcc_id,      supply_id,      employee_id,      storage_id,
accounting_date,      quantity,      last_movement_date      FROM
product_accounting WHERE productAcc_id = @id", conn))
                {
                    cmd.Parameters.AddWithValue("id", id);
                    using (var reader = cmd.ExecuteReader())
                    {
                        if (reader.Read())
                        {
                            return new ProductAccounting
```

```
                        {
                        ProductAccId = reader.GetInt32(0),
                                SupplyId = reader.GetInt32(1),
                                EmployeeId = reader.GetInt32(2),
                                StorageId = reader.GetInt32(3),
                        AccountingDate = reader.GetDateTime(4),
                                Quantity = reader.GetInt32(5),
                        LastMovementDate                        =
reader.IsDBNull(6) ? null : reader.GetDateTime(6)
                        };
                        }
                        return null;
                    }
                }
            }
        }

        public void Add(ProductAccounting productAccounting)
        {
            using (var conn = _dbConnection.GetConnection())
            {
                conn.Open();
                using (var cmd = new NpgsqlCommand(
                    "INSERT INTO product_accounting (supply_id,
employee_id,      storage_id,      accounting_date,      quantity,
last_movement_date)     VALUES     (@supply_id,     @employee_id,
@storage_id,  @accounting_date,  @quantity,  @last_movement_date)
RETURNING productAcc_id", conn))
                {
                    cmd.Parameters.AddWithValue("supply_id",
productAccounting.SupplyId);
                    cmd.Parameters.AddWithValue("employee_id",
productAccounting.EmployeeId);
                    cmd.Parameters.AddWithValue("storage_id",
productAccounting.StorageId);
                cmd.Parameters.AddWithValue("accounting_date",
productAccounting.AccountingDate);
```

```
                    cmd.Parameters.AddWithValue("quantity",
productAccounting.Quantity);
                cmd.Parameters.AddWithValue("last_movement_date",
(object)productAccounting.LastMovementDate ?? DBNull.Value);
                    productAccounting.ProductAccId              =
(int)cmd.ExecuteScalar();
                }
            }
        }

        public void Update(ProductAccounting productAccounting)
        {
            using (var conn = _dbConnection.GetConnection())
            {
                conn.Open();
                using (var cmd = new NpgsqlCommand(
                    "UPDATE product_accounting SET supply_id =
@supply_id, employee_id = @employee_id, storage_id = @storage_id,
accounting_date = @accounting_date, quantity = @quantity,
last_movement_date = @last_movement_date WHERE productAcc_id =
@id", conn))
                {
                    cmd.Parameters.AddWithValue("id",
productAccounting.ProductAccId);
                    cmd.Parameters.AddWithValue("supply_id",
productAccounting.SupplyId);
                    cmd.Parameters.AddWithValue("employee_id",
productAccounting.EmployeeId);
                    cmd.Parameters.AddWithValue("storage_id",
productAccounting.StorageId);
                cmd.Parameters.AddWithValue("accounting_date",
productAccounting.AccountingDate);
                    cmd.Parameters.AddWithValue("quantity",
productAccounting.Quantity);
                cmd.Parameters.AddWithValue("last_movement_date",
(object)productAccounting.LastMovementDate ?? DBNull.Value);
                    cmd.ExecuteNonQuery();
                }
```

```
            }
        }

        public void Delete(int id)
        {
            using (var conn = _dbConnection.GetConnection())
            {
                conn.Open();
                using (var cmd = new NpgsqlCommand("DELETE FROM
product_accounting WHERE productAcc_id = @id", conn))
                {
                    cmd.Parameters.AddWithValue("id", id);
                    cmd.ExecuteNonQuery();
                }
            }
        }
    }
}
```

Листинг файла ProductRepository.cs:

```
using Npgsql;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using Domain.Interfaces;
using Domain.Models;
using NpgsqlTypes;

namespace Infrastructure.Repositories
{
    public class ProductRepository : IProductRepository
    {
```

```
        private readonly DatabaseConnection _dbConnection;


      public ProductRepository(DatabaseConnection dbConnection)
        {
            _dbConnection = dbConnection ?? throw new
ArgumentNullException(nameof(dbConnection));
        }


        public List<Product> GetAll()
        {
            var products = new List<Product>();
            using (var conn = _dbConnection.GetConnection())
            {
                conn.Open();
            using (var cmd = new NpgsqlCommand("SELECT product_id,
name, expiry_date, product_type, is_active, photo FROM product",
conn))
                using (var reader = cmd.ExecuteReader())
                {
                    while (reader.Read())
                    {
                        products.Add(new Product
                        {
                            ProductId = reader.GetInt32(0),
                            Name = reader.GetString(1),
                     ExpiryDate = reader.IsDBNull(2) ?
null : reader.GetDateTime(2),
                            ProductType =
(ProductType)Enum.Parse(typeof(ProductType), reader.GetString(3),
true),
                            IsActive = reader.GetBoolean(4),
                            Photo = reader.IsDBNull(5) ? null :
(byte[])reader.GetValue(5)
                        });
                    }
                }
```

```
            }
            return products;
        }

        public Product GetById(int id)
        {
            using (var conn = _dbConnection.GetConnection())
            {
                conn.Open();
            using (var cmd = new NpgsqlCommand("SELECT product_id, name, expiry_date, product_type, is_active, photo FROM product WHERE product_id = @id", conn))
                {
                    cmd.Parameters.AddWithValue("id", id);
                    using (var reader = cmd.ExecuteReader())
                    {
                        if (reader.Read())
                        {
                            return new Product
                            {
                                ProductId = reader.GetInt32(0),
                                Name = reader.GetString(1),
                                ExpiryDate                          = reader.IsDBNull(2) ? null : reader.GetDateTime(2),
                                ProductType                         = (ProductType)Enum.Parse(typeof(ProductType), reader.GetString(3), true),
                                IsActive = reader.GetBoolean(4),
                        Photo     =     reader.IsDBNull(5)     ? null : (byte[])reader.GetValue(5)
                            };
                        }
                        return null;
                    }
                }
```

```
            }
        }

        public void Add(Product product)
        {
            using (var conn = _dbConnection.GetConnection())
            {
                conn.Open();
                using (var cmd = new NpgsqlCommand(
                    "INSERT INTO product (name, expiry_date, product_type, is_active, photo) VALUES (@name, @expiry_date, @product_type, @is_active, @photo) RETURNING product_id", conn))
                {
                cmd.Parameters.AddWithValue("name", product.Name);
                    cmd.Parameters.AddWithValue("expiry_date", (object)product.ExpiryDate ?? DBNull.Value);
                    cmd.Parameters.Add(new NpgsqlParameter("product_type", NpgsqlDbType.Unknown) { Value = product.ProductType.ToString(), DataTypeName = "product_type" });
                    cmd.Parameters.AddWithValue("is_active", product.IsActive);
                    cmd.Parameters.AddWithValue("photo", (object)product.Photo ?? DBNull.Value);
                product.ProductId = (int)cmd.ExecuteScalar();
                }
            }
        }

        public void Update(Product product)
        {
            using (var conn = _dbConnection.GetConnection())
            {
                conn.Open();
                using (var cmd = new NpgsqlCommand(
               "UPDATE product SET name = @name, expiry_date = @expiry_date, product_type = @product_type, is_active = @is_active, photo = @photo WHERE product_id = @id", conn))
```

```
                {
                    cmd.Parameters.AddWithValue("id",
product.ProductId);
                cmd.Parameters.AddWithValue("name", product.Name);
                    cmd.Parameters.AddWithValue("expiry_date",
(object)product.ExpiryDate ?? DBNull.Value);
                    cmd.Parameters.Add(new
NpgsqlParameter("product_type", NpgsqlDbType.Unknown) { Value =
product.ProductType.ToString(), DataTypeName = "product_type" });
                    cmd.Parameters.AddWithValue("is_active",
product.IsActive);
                    cmd.Parameters.AddWithValue("photo",
(object)product.Photo ?? DBNull.Value);
                    cmd.ExecuteNonQuery();
                }
            }
        }

        public void Delete(int id)
        {
            using (var conn = _dbConnection.GetConnection())
            {
                conn.Open();
                using (var cmd = new NpgsqlCommand("DELETE FROM
product WHERE product_id = @id", conn))
                {
                    cmd.Parameters.AddWithValue("id", id);
                    cmd.ExecuteNonQuery();
                }
            }
        }
    }
}
```

Листинг файла StorageZoneRepository.cs:

```
using System;
```

```
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using Domain.Interfaces;
using Domain.Models;
using Npgsql;
using NpgsqlTypes;

namespace Infrastructure.Repositories
{
    public class StorageZoneRepository : IStorageZoneRepository
    {
        private readonly DatabaseConnection _dbConnection;

        public StorageZoneRepository(DatabaseConnection
dbConnection)
        {
            _dbConnection = dbConnection ?? throw new
ArgumentNullException(nameof(dbConnection));
        }

        public List<StorageZone> GetAll()
        {
            var storageZones = new List<StorageZone>();
            using (var conn = _dbConnection.GetConnection())
            {
                conn.Open();
            using (var cmd = new NpgsqlCommand("SELECT storage_id,
warehouse_id, capacity, zone_type, zone_name FROM storage_zone",
conn))
                using (var reader = cmd.ExecuteReader())
                {
                    while (reader.Read())
                    {
```

```
                        storageZones.Add(new StorageZone
                        {
                            StorageId = reader.GetInt32(0),
                            WarehouseId = reader.GetInt32(1),
                            Capacity = reader.GetInt32(2),
                            ZoneType                                =
(ZoneType)Enum.Parse(typeof(ZoneType),       reader.GetString(3),
true),
                            ZoneName = reader.GetString(4)
                        });
                    }
                }
            }
            return storageZones;
        }

        public StorageZone GetById(int id)
        {
            using (var conn = _dbConnection.GetConnection())
            {
                conn.Open();
            using (var cmd = new NpgsqlCommand("SELECT storage_id,
warehouse_id, capacity, zone_type, zone_name FROM storage_zone
WHERE storage_id = @id", conn))
                {
                    cmd.Parameters.AddWithValue("id", id);
                    using (var reader = cmd.ExecuteReader())
                    {
                        if (reader.Read())
                        {
                            return new StorageZone
                            {
                                StorageId = reader.GetInt32(0),
                        WarehouseId = reader.GetInt32(1),
                                Capacity = reader.GetInt32(2),
```

```
                                ZoneType                        =
(ZoneType)Enum.Parse(typeof(ZoneType),      reader.GetString(3),
true),
                                ZoneName = reader.GetString(4)
                            };
                        }
                        return null;
                    }
                }
            }

        public void Add(StorageZone storageZone)
        {
            using (var conn = _dbConnection.GetConnection())
            {
                conn.Open();
                using (var cmd = new NpgsqlCommand(
               "INSERT INTO storage_zone (warehouse_id, capacity,
zone_type,   zone_name)   VALUES   (@warehouse_id,   @capacity,
@zone_type, @zone_name) RETURNING storage_id", conn))
                {
                    cmd.Parameters.AddWithValue("warehouse_id",
storageZone.WarehouseId);
                    cmd.Parameters.AddWithValue("capacity",
storageZone.Capacity);
                    cmd.Parameters.Add(new
NpgsqlParameter("zone_type",  NpgsqlDbType.Unknown)  {  Value  =
storageZone.ZoneType.ToString(), DataTypeName = "zone_type" });
                    cmd.Parameters.AddWithValue("zone_name",
storageZone.ZoneName);
                storageZone.StorageId = (int)cmd.ExecuteScalar();
                }
            }
        }

        public void Update(StorageZone storageZone)
```

```
        {
            using (var conn = _dbConnection.GetConnection())
            {
                conn.Open();
                using (var cmd = new NpgsqlCommand(
                    "UPDATE storage_zone SET warehouse_id = @warehouse_id, capacity = @capacity, zone_type = @zone_type, zone_name = @zone_name WHERE storage_id = @id", conn))
                {
                    cmd.Parameters.AddWithValue("id", storageZone.StorageId);
                    cmd.Parameters.AddWithValue("warehouse_id", storageZone.WarehouseId);
                    cmd.Parameters.AddWithValue("capacity", storageZone.Capacity);
                    cmd.Parameters.Add(new NpgsqlParameter("zone_type", NpgsqlDbType.Unknown) { Value = storageZone.ZoneType.ToString(), DataTypeName = "zone_type" });
                    cmd.Parameters.AddWithValue("zone_name", storageZone.ZoneName);
                    cmd.ExecuteNonQuery();
                }
            }
        }

        public void Delete(int id)
        {
            using (var conn = _dbConnection.GetConnection())
            {
                conn.Open();
                using (var cmd = new NpgsqlCommand("DELETE FROM storage_zone WHERE storage_id = @id", conn))
                {
                    cmd.Parameters.AddWithValue("id", id);
                    cmd.ExecuteNonQuery();
                }
            }
```

```
        }
    }
}
```

Листинг файла SupplierRepository.cs:

```
using Npgsql;
using System;
using Domain.Interfaces;
using Domain.Models;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Infrastructure.Repositories
{
    public class SupplierRepository : ISupplierRepository
    {
        private readonly DatabaseConnection _dbConnection;

      public SupplierRepository(DatabaseConnection dbConnection)
        {
            _dbConnection = dbConnection ?? throw new
ArgumentNullException(nameof(dbConnection));
        }

        public List<Supplier> GetAll()
        {
            var suppliers = new List<Supplier>();
            using (var conn = _dbConnection.GetConnection())
            {
                conn.Open();
                using (var cmd = new NpgsqlCommand("SELECT
supplier_id, company_name, contact_person, phone, address FROM
supplier", conn))
```

```
                using (var reader = cmd.ExecuteReader())
                {
                    while (reader.Read())
                    {
                        suppliers.Add(new Supplier
                        {
                            SupplierId = reader.GetInt32(0),
                            CompanyName = reader.GetString(1),
                            ContactPerson = reader.IsDBNull(2) ? 
null : reader.GetString(2),
                            Phone = reader.IsDBNull(3) ? null : 
reader.GetString(3),
                            Address = reader.IsDBNull(4) ? 
null : reader.GetString(4)
                        });
                    }
                }
            }
            return suppliers;
        }

        public Supplier GetById(int id)
        {
            using (var conn = _dbConnection.GetConnection())
            {
                conn.Open();
                using (var cmd = new NpgsqlCommand("SELECT 
supplier_id, company_name, contact_person, phone, address FROM 
supplier WHERE supplier_id = @id", conn))
                {
                    cmd.Parameters.AddWithValue("id", id);
                    using (var reader = cmd.ExecuteReader())
                    {
                        if (reader.Read())
                        {
```

```
                            return new Supplier
                            {
                                SupplierId = reader.GetInt32(0),
                            CompanyName = reader.GetString(1),
                            ContactPerson                                =
reader.IsDBNull(2) ? null : reader.GetString(2),
                            Phone     =     reader.IsDBNull(3)     ?
null : reader.GetString(3),
                            Address     =     reader.IsDBNull(4)     ?
null : reader.GetString(4)
                            };
                            }
                            return null;
                        }
                    }
                }
            }

            public void Add(Supplier supplier)
            {
                using (var conn = _dbConnection.GetConnection())
                {
                    conn.Open();
                    using (var cmd = new NpgsqlCommand(
                        "INSERT    INTO    supplier    (company_name,
contact_person,    phone,    address)    VALUES    (@company_name,
@contact_person, @phone, @address) RETURNING supplier_id", conn))
                    {
                        cmd.Parameters.AddWithValue("company_name",
supplier.CompanyName);
                    cmd.Parameters.AddWithValue("contact_person",
(object)supplier.ContactPerson ?? DBNull.Value);
                        cmd.Parameters.AddWithValue("phone",
(object)supplier.Phone ?? DBNull.Value);
                        cmd.Parameters.AddWithValue("address",
(object)supplier.Address ?? DBNull.Value);
                    supplier.SupplierId = (int)cmd.ExecuteScalar();
```

```
            }
        }
    }

    public void Update(Supplier supplier)
    {
        using (var conn = _dbConnection.GetConnection())
        {
            conn.Open();
            using (var cmd = new NpgsqlCommand(
           "UPDATE supplier SET company_name = @company_name,
contact_person = @contact_person, phone = @phone, address =
@address WHERE supplier_id = @id", conn))
            {
                cmd.Parameters.AddWithValue("id",
supplier.SupplierId);
                cmd.Parameters.AddWithValue("company_name",
supplier.CompanyName);
           cmd.Parameters.AddWithValue("contact_person",
(object)supplier.ContactPerson ?? DBNull.Value);
                cmd.Parameters.AddWithValue("phone",
(object)supplier.Phone ?? DBNull.Value);
                cmd.Parameters.AddWithValue("address",
(object)supplier.Address ?? DBNull.Value);
                cmd.ExecuteNonQuery();
            }
        }
    }

    public void Delete(int id)
    {
        using (var conn = _dbConnection.GetConnection())
        {
            conn.Open();
            using (var cmd = new NpgsqlCommand("DELETE FROM
supplier WHERE supplier_id = @id", conn))
```

```
                {
                    cmd.Parameters.AddWithValue("id", id);
                    cmd.ExecuteNonQuery();
                }
            }
        }
    }
}
```

Листинг файла SupplyRepository.cs:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using Npgsql;
using Domain.Interfaces;
using Domain.Models;

namespace Infrastructure.Repositories
{
    public class SupplyRepository : ISupplyRepository
    {
        private readonly DatabaseConnection _dbConnection;

        public SupplyRepository(DatabaseConnection dbConnection)
        {
            _dbConnection = dbConnection ?? throw new
ArgumentNullException(nameof(dbConnection));
        }

        public List<Supply> GetAll()
        {
            var supplies = new List<Supply>();
```

```
            using (var conn = _dbConnection.GetConnection())
            {
                conn.Open();
            using (var cmd = new NpgsqlCommand("SELECT supply_id,
product_id,  supplier_id,  supply_date,  quantity  FROM  supply",
conn))
                using (var reader = cmd.ExecuteReader())
                {
                    while (reader.Read())
                    {
                        supplies.Add(new Supply
                        {
                            SupplyId = reader.GetInt32(0),
                            ProductId = reader.GetInt32(1),
                            SupplierId = reader.GetInt32(2),
                            SupplyDate = reader.GetDateTime(3),
                            Quantity = reader.GetInt32(4)
                        });
                    }
                }
            }
            return supplies;
        }


        public Supply GetById(int id)
        {
            using (var conn = _dbConnection.GetConnection())
            {
                conn.Open();
            using (var cmd = new NpgsqlCommand("SELECT supply_id,
product_id, supplier_id, supply_date, quantity FROM supply WHERE
supply_id = @id", conn))
                {
                    cmd.Parameters.AddWithValue("id", id);
                    using (var reader = cmd.ExecuteReader())
```

```
                    {
                        if (reader.Read())
                        {
                            return new Supply
                            {
                                SupplyId = reader.GetInt32(0),
                                ProductId = reader.GetInt32(1),
                                SupplierId = reader.GetInt32(2),
                        SupplyDate = reader.GetDateTime(3),
                                Quantity = reader.GetInt32(4)
                            };
                        }
                        return null;
                    }
                }
            }
        }

        public void Add(Supply supply)
        {
            using (var conn = _dbConnection.GetConnection())
            {
                conn.Open();
                using (var cmd = new NpgsqlCommand(
               "INSERT INTO supply (product_id, supplier_id,
supply_date,  quantity)  VALUES  (@product_id,  @supplier_id,
@supply_date, @quantity) RETURNING supply_id", conn))
                {
                    cmd.Parameters.AddWithValue("product_id",
supply.ProductId);
                    cmd.Parameters.AddWithValue("supplier_id",
supply.SupplierId);
                    cmd.Parameters.AddWithValue("supply_date",
supply.SupplyDate);
```

```
                    cmd.Parameters.AddWithValue("quantity",
supply.Quantity);
                    supply.SupplyId = (int)cmd.ExecuteScalar();
                }
            }
        }

        public void Update(Supply supply)
        {
            using (var conn = _dbConnection.GetConnection())
            {
                conn.Open();
                using (var cmd = new NpgsqlCommand(
                    "UPDATE supply SET product_id = @product_id,
supplier_id = @supplier_id, supply_date = @supply_date, quantity
= @quantity WHERE supply_id = @id", conn))
                {
                    cmd.Parameters.AddWithValue("id",
supply.SupplyId);
                    cmd.Parameters.AddWithValue("product_id",
supply.ProductId);
                    cmd.Parameters.AddWithValue("supplier_id",
supply.SupplierId);
                    cmd.Parameters.AddWithValue("supply_date",
supply.SupplyDate);
                    cmd.Parameters.AddWithValue("quantity",
supply.Quantity);
                    cmd.ExecuteNonQuery();
                }
            }
        }

        public void Delete(int id)
        {
            using (var conn = _dbConnection.GetConnection())
            {
```

```
                conn.Open();
                using (var cmd = new NpgsqlCommand("DELETE FROM
supply WHERE supply_id = @id", conn))
                {
                    cmd.Parameters.AddWithValue("id", id);
                    cmd.ExecuteNonQuery();
                }
            }
        }
    }
}
```

Листинг файла WarehouseRepository.cs:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using Domain.Interfaces;
using Domain.Models;
using Npgsql;

namespace Infrastructure.Repositories
{
    public class WarehouseRepository : IWarehouseRepository
    {
        private readonly DatabaseConnection _dbConnection;

      public WarehouseRepository(DatabaseConnection dbConnection)
        {
            _dbConnection = dbConnection ?? throw new
ArgumentNullException(nameof(dbConnection));
        }
```

```
        public List<Warehouse> GetAll()
        {
            var warehouses = new List<Warehouse>();
            using (var conn = _dbConnection.GetConnection())
            {
                conn.Open();
                using (var cmd = new NpgsqlCommand("SELECT warehouse_id, name, address FROM warehouse", conn))
                using (var reader = cmd.ExecuteReader())
                {
                    while (reader.Read())
                    {
                        warehouses.Add(new Warehouse
                        {
                            WarehouseId = reader.GetInt32(0),
                            Name = reader.GetString(1),
                            Address = reader.GetString(2)
                        });
                    }
                }
            }
            return warehouses;
        }

        public Warehouse GetById(int id)
        {
            using (var conn = _dbConnection.GetConnection())
            {
                conn.Open();
                using (var cmd = new NpgsqlCommand("SELECT warehouse_id, name, address FROM warehouse WHERE warehouse_id = @id", conn))
                {
                    cmd.Parameters.AddWithValue("id", id);
```

```
                    using (var reader = cmd.ExecuteReader())
                    {
                        if (reader.Read())
                        {
                            return new Warehouse
                            {
                        WarehouseId = reader.GetInt32(0),
                                Name = reader.GetString(1),
                                Address = reader.GetString(2)
                            };
                        }
                        return null;
                    }
                }
            }

        public void Add(Warehouse warehouse)
        {
            using (var conn = _dbConnection.GetConnection())
            {
                conn.Open();
                using (var cmd = new NpgsqlCommand(
               "INSERT INTO warehouse (name, address) VALUES
(@name, @address) RETURNING warehouse_id", conn))
                {
                    cmd.Parameters.AddWithValue("name",
warehouse.Name);
                    cmd.Parameters.AddWithValue("address",
warehouse.Address);
               warehouse.WarehouseId = (int)cmd.ExecuteScalar();
                }
            }
        }
```

```
        public void Update(Warehouse warehouse)
        {
            using (var conn = _dbConnection.GetConnection())
            {
                conn.Open();
                using (var cmd = new NpgsqlCommand(
                    "UPDATE warehouse SET name = @name, address
= @address WHERE warehouse_id = @id", conn))
                {
                    cmd.Parameters.AddWithValue("id",
warehouse.WarehouseId);
                    cmd.Parameters.AddWithValue("name",
warehouse.Name);
                    cmd.Parameters.AddWithValue("address",
warehouse.Address);
                    cmd.ExecuteNonQuery();
                }
            }
        }

        public void Delete(int id)
        {
            using (var conn = _dbConnection.GetConnection())
            {
                conn.Open();
                using (var cmd = new NpgsqlCommand("DELETE FROM
warehouse WHERE warehouse_id = @id", conn))
                {
                    cmd.Parameters.AddWithValue("id", id);
                    cmd.ExecuteNonQuery();
                }
            }
        }
    }
```

```
}
```

Листинг проекта UI

Листинг файла AboutForm.cs:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Diagnostics;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace UI
{
    public partial class AboutForm : Form
    {
        public AboutForm()
        {
            InitializeComponent();
        }

        private void btnClose_Click(object sender, EventArgs e)
        {
            this.Close();
        }

        private    void    lnkGitHub_LinkClicked(object    sender,
LinkLabelLinkClickedEventArgs e)
        {
            try
            {
```

```
                var psi = new ProcessStartInfo
                {
                    FileName     =     "https://github.com/markld-
ui/DataBase",
                    UseShellExecute = true
                };

                System.Diagnostics.Process.Start(psi);
            }
            catch (Exception ex)
            {
                MessageBox.Show($"Не   удалось   открыть   ссылку:
{ex.Message}",          "Ошибка",          MessageBoxButtons.OK,
MessageBoxIcon.Error);
            }
        }
    }
}
```

Листинг файла App.config:

```
<?xml version="1.0" encoding="utf-8"?>
<configuration>
    <configSections>
        <sectionGroup                              name="userSettings"
type="System.Configuration.UserSettingsGroup,             System,
Version=4.0.0.0,                                  Culture=neutral,
PublicKeyToken=b77a5c561934e089">
            <section                               name="UI.Settings"
type="System.Configuration.ClientSettingsSection,         System,
Version=4.0.0.0,                                  Culture=neutral,
PublicKeyToken=b77a5c561934e089"
allowExeDefinition="MachineToLocalUser" requirePermission="false"
/>
        </sectionGroup>
    </configSections>
    <userSettings>
```

```
        <UI.Settings>
         <setting name="MainFormLocation" serializeAs="String">
                <value>0, 0</value>
            </setting>
            <setting name="MainFormSize" serializeAs="String">
                <value>800, 600</value>
            </setting>
            <setting name="MainFormState" serializeAs="String">
                <value>Normal</value>
            </setting>
         <setting name="MainFormMaximized" serializeAs="String">
                <value>False</value>
            </setting>
        </UI.Settings>
    </userSettings>
    <connectionStrings>
        <add                              name="WarehouseConnection"
connectionString="Host=localhost;Port=5432;Database=ORBD;Usernam
e=postgres;Password=Hjvfy3105" providerName="Npgsql" />
    </connectionStrings>
</configuration>
```

Листинг файла EmployeeForm.cs:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using Domain.Interfaces;
using Domain.Models;
```

```
namespace UI
{
    public partial class EmployeeForm : Form
    {
      private readonly IEmployeeRepository _employeeRepository;
        private Employee _selectedEmployee;
        private BindingSource _bindingSource;
        private BindingNavigator _bindingNavigator;

        public EmployeeForm()
        {
            InitializeComponent();
            _employeeRepository                                    =
Application.AppContext.Instance.EmployeeRepository;
            _bindingSource = new BindingSource();

            _bindingNavigator                =                 new
BindingNavigator(_bindingSource);
            _bindingNavigator.Dock = DockStyle.Top;
            this.Controls.Add(_bindingNavigator);

         _bindingNavigator.MoveFirstItem.ToolTipText  =  "Перейти
к первой записи";
            _bindingNavigator.MovePreviousItem.ToolTipText      =
"Перейти к предыдущей записи";
         _bindingNavigator.MoveNextItem.ToolTipText   =   "Перейти
к следующей записи";
         _bindingNavigator.MoveLastItem.ToolTipText   =   "Перейти
к последней записи";
            _bindingNavigator.AddNewItem.ToolTipText  =  "Добавить
новую запись";
            _bindingNavigator.DeleteItem.ToolTipText  =  "Удалить
текущую запись";
         _bindingNavigator.PositionItem.ToolTipText   =   "Текущая
позиция";
```

```
            _bindingNavigator.CountItem.ToolTipText    =    "Общее
количество записей";
        }

        private  void  EmployeeForm_Load(object  sender,  EventArgs
e)
        {
            LoadEmployees();
        }

        private void LoadEmployees()
        {
            try
            {
                var employees = _employeeRepository.GetAll();
                _bindingSource.DataSource = employees;
            dataGridViewEmployees.DataSource = _bindingSource;


dataGridViewEmployees.Columns["EmployeeId"].HeaderText = "ID";
            dataGridViewEmployees.Columns["FullName"].HeaderText
= "ФИО";
            dataGridViewEmployees.Columns["Position"].HeaderText
= "Должность";
            dataGridViewEmployees.Columns["Phone"].HeaderText
= "Телефон";
                dataGridViewEmployees.Columns["Photo"].Visible  =
false;
            }
            catch (Exception ex)
            {
                MessageBox.Show($"Ошибка  загрузки  сотрудников:
{ex.Message}",          "Ошибка",          MessageBoxButtons.OK,
MessageBoxIcon.Error);
            }
        }
```

```
      private void dataGridViewEmployees_SelectionChanged(object
sender, EventArgs e)
        {
            if (dataGridViewEmployees.SelectedRows.Count > 0)
            {
                var                 selectedRow                 =
dataGridViewEmployees.SelectedRows[0];
                _selectedEmployee = selectedRow.DataBoundItem as
Employee;
                if (_selectedEmployee != null)
                {
               txtFullName.Text = _selectedEmployee.FullName;
               txtPosition.Text = _selectedEmployee.Position;
                    txtPhone.Text = _selectedEmployee.Phone;
                    if (_selectedEmployee.Photo != null)
                    {
                        using      (var      ms      =      new
System.IO.MemoryStream(_selectedEmployee.Photo))
                        {
                     picPhoto.Image = Image.FromStream(ms);
                        }
                    }
                    else
                    {
                        picPhoto.Image = null;
                    }
                }
            }
        }

      private void btnUploadPhoto_Click(object sender, EventArgs
e)
        {
            using (var openFileDialog = new OpenFileDialog())
            {
```

```
                openFileDialog.Filter = "Image Files|*.jpg;*.jpeg;*.png;*.bmp";
            if (openFileDialog.ShowDialog() == DialogResult.OK)
                {
                    picPhoto.Image = Image.FromFile(openFileDialog.FileName);
                }
            }
        }

        private void btnAdd_Click(object sender, EventArgs e)
        {
            try
            {
                if (!ValidateInput()) return;

                var employee = new Employee
                {
                    FullName = txtFullName.Text,
                    Position = txtPosition.Text,
                    Phone = txtPhone.Text,
                    Photo = picPhoto.Image != null ? ImageToByteArray(picPhoto.Image) : null
                };

                _employeeRepository.Add(employee);
                LoadEmployees();
                ClearInputs();
            }
            catch (Exception ex)
            {
                MessageBox.Show($"Ошибка добавления сотрудника: {ex.Message}", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);
            }
```

```
        }

        private void btnUpdate_Click(object sender, EventArgs e)
        {
            try
            {
                if (_selectedEmployee == null)
                {
                    MessageBox.Show("Выберите сотрудника для обновления.", "Предупреждение", MessageBoxButtons.OK, MessageBoxIcon.Warning);
                    return;
                }

                if (!ValidateInput()) return;

                _selectedEmployee.FullName = txtFullName.Text;
                _selectedEmployee.Position = txtPosition.Text;
                _selectedEmployee.Phone = txtPhone.Text;
                _selectedEmployee.Photo = picPhoto.Image != null ? ImageToByteArray(picPhoto.Image) : null;

                _employeeRepository.Update(_selectedEmployee);
                LoadEmployees();
                ClearInputs();
            }
            catch (Exception ex)
            {
                MessageBox.Show($"Ошибка обновления сотрудника: {ex.Message}", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);
            }
        }

        private void btnDelete_Click(object sender, EventArgs e)
```

```
        {
            try
            {
                if (_selectedEmployee == null)
                {
                    MessageBox.Show("Выберите сотрудника для
удаления.", "Предупреждение", MessageBoxButtons.OK,
MessageBoxIcon.Warning);
                    return;
                }

            if (MessageBox.Show($"Вы уверены, что хотите удалить
сотрудника '{_selectedEmployee.FullName}'?", "Подтверждение",
MessageBoxButtons.YesNo, MessageBoxIcon.Question) ==
DialogResult.Yes)
                {

_employeeRepository.Delete(_selectedEmployee.EmployeeId);
                    LoadEmployees();
                    ClearInputs();
                }
            }
            catch (Exception ex)
            {
                MessageBox.Show($"Ошибка удаления сотрудника:
{ex.Message}", "Ошибка", MessageBoxButtons.OK,
MessageBoxIcon.Error);
            }
        }

        private bool ValidateInput()
        {
            if (string.IsNullOrWhiteSpace(txtFullName.Text))
            {
            MessageBox.Show("Введите ФИО сотрудника.", "Ошибка",
MessageBoxButtons.OK, MessageBoxIcon.Warning);
                return false;
```

```
            }

            if (!string.IsNullOrWhiteSpace(txtPhone.Text))
            {
                if
(!System.Text.RegularExpressions.Regex.IsMatch(txtPhone.Text,
@"^\+?\d{10,15}$"))
                {
                    MessageBox.Show("Введите корректный номер
телефона.", "Ошибка", MessageBoxButtons.OK,
MessageBoxIcon.Warning);
                    return false;
                }
            }

            return true;
        }

        private void ClearInputs()
        {
            txtFullName.Clear();
            txtPosition.Clear();
            txtPhone.Clear();
            picPhoto.Image = null;
            _selectedEmployee = null;
        }

        private byte[] ImageToByteArray(Image image)
        {
            using (var ms = new System.IO.MemoryStream())
            {
                image.Save(ms, image.RawFormat);
                return ms.ToArray();
            }
        }
```

```
    }
}
```

Листинг файла MainForm.cs:

```
using System;
using System.Windows.Forms;
using Application;
using Infrastructure;

namespace UI
{
    public partial class MainForm : Form
    {
        public MainForm()
        {
            InitializeComponent();
            LoadSettings();
            TestDatabaseConnection()
        }

        private void TestDatabaseConnection()
        {
            var dbConnection = Application.AppContext.Instance.DatabaseConnection;
            if (dbConnection.TestConnection())
            {
            MessageBox.Show("Подключение к базе данных успешно!", "Успех", MessageBoxButtons.OK, MessageBoxIcon.Information);
            }
            else
            {
                MessageBox.Show("Не удалось подключиться к базе данных. Проверьте настройки.", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);
            }
```

```
        }

      private void productsToolStripMenuItem_Click(object sender,
EventArgs e)
        {
            var form = new ProductForm();
            form.ShowDialog();
        }

        private   void   suppliersToolStripMenuItem_Click(object
sender, EventArgs e)
        {
            var form = new SupplierForm();
            form.ShowDialog();
        }

      private void suppliesToolStripMenuItem_Click(object sender,
EventArgs e)
        {
            var form = new SupplyForm();
            form.ShowDialog();
        }

        private   void   employeesToolStripMenuItem_Click(object
sender, EventArgs e)
        {
            var form = new EmployeeForm();
            form.ShowDialog();
        }

        private                                              void
productAccountingToolStripMenuItem_Click(object sender, EventArgs
e)
        {
            var form = new ProductAccountingForm();
```

```
            form.ShowDialog();
        }

        private void storageZonesToolStripMenuItem_Click(object
sender, EventArgs e)
        {
            var form = new StorageZoneForm();
            form.ShowDialog();
        }

        private void warehousesToolStripMenuItem_Click(object
sender, EventArgs e)
        {
            var form = new WarehouseForm();
            form.ShowDialog();
        }

        private void aboutToolStripMenuItem_Click(object sender,
EventArgs e)
        {
            var form = new AboutForm();
            form.ShowDialog();
        }

        private void exitToolStripMenuItem_Click(object sender,
EventArgs e)
        {
            this.Close();
        }

        private void MainForm_FormClosing(object sender,
FormClosingEventArgs e)
        {
            if (this.WindowState == FormWindowState.Maximized)
            {
```

```
            Settings.Default.MainFormMaximized = true;
        }
        else
        {
            Settings.Default.MainFormMaximized = false;
        Settings.Default.MainFormLocation = this.Location;
            Settings.Default.MainFormSize = this.Size;
        }
        Settings.Default.Save();

        e.Cancel = MessageBox.Show("Вы хотите закрыть программу?", "Внимание", MessageBoxButtons.YesNo, MessageBoxIcon.Question) != DialogResult.Yes;
    }

    private void MainForm_FormClosed(object sender, FormClosedEventArgs e)
    {
        SaveSettings();
    }

    private void LoadSettings()
    {
        this.Location = Settings.Default.MainFormLocation;
        this.Size = Settings.Default.MainFormSize;

        if (Enum.TryParse<FormWindowState>(Settings.Default.MainFormState, out var state))
        {
            this.WindowState = state;
        }
        else
        {
            this.WindowState = FormWindowState.Normal;
```

```
            }
        }

        private void SaveSettings()
        {
            if (this.WindowState == FormWindowState.Normal)
            {
            Settings.Default.MainFormLocation = this.Location;
                Settings.Default.MainFormSize = this.Size;
            }

            Settings.Default.MainFormState                          =
this.WindowState.ToString();
            Settings.Default.Save();
        }

        private MenuStrip menuStripMain;
        private ContextMenuStrip contextMenuStripMain;
        private ToolStrip toolStripMain;
    }
}
```

Листинг файла ProductAccountingForm.cs:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using Domain.Interfaces;
```

```
using Domain.Models;


namespace UI
{
    public partial class ProductAccountingForm : Form
    {
        private readonly IProductAccountingRepository
_productAccountingRepository;
        private readonly ISupplyRepository _supplyRepository;
      private readonly IEmployeeRepository _employeeRepository;
        private readonly IStorageZoneRepository
_storageZoneRepository;
        private ProductAccounting _selectedProductAccounting;
        private BindingSource _bindingSource;
        private BindingNavigator _bindingNavigator;


        public ProductAccountingForm()
        {
            InitializeComponent();
            _productAccountingRepository =
Application.AppContext.Instance.ProductAccountingRepository;
            _supplyRepository =
Application.AppContext.Instance.SupplyRepository;
            _employeeRepository =
Application.AppContext.Instance.EmployeeRepository;
            _storageZoneRepository =
Application.AppContext.Instance.StorageZoneRepository;
            _bindingSource = new BindingSource();


            _bindingNavigator = new
BindingNavigator(_bindingSource);
            _bindingNavigator.Dock = DockStyle.Top;
            this.Controls.Add(_bindingNavigator);


         _bindingNavigator.MoveFirstItem.ToolTipText = "Перейти
к первой записи";
```

```
            _bindingNavigator.MovePreviousItem.ToolTipText      =
"Перейти к предыдущей записи";

          _bindingNavigator.MoveNextItem.ToolTipText   =   "Перейти
к следующей записи";

          _bindingNavigator.MoveLastItem.ToolTipText   =   "Перейти
к последней записи";

            _bindingNavigator.AddNewItem.ToolTipText  =  "Добавить
новую запись";

            _bindingNavigator.DeleteItem.ToolTipText  =  "Удалить
текущую запись";

          _bindingNavigator.PositionItem.ToolTipText   =   "Текущая
позиция";

            _bindingNavigator.CountItem.ToolTipText    =    "Общее
количество записей";

        }


        private  void  ProductAccountingForm_Load(object  sender,
EventArgs e)

        {

            LoadProductAccountings();

            LoadSupplies();

            LoadEmployees();

            LoadStorageZones();

            dtpLastMovementDate.Enabled = false;

        }


        private void LoadProductAccountings()

        {

            try

            {

                var              productAccountings              =
_productAccountingRepository.GetAll();

                _bindingSource.DataSource = productAccountings;

                dataGridViewProductAccountings.DataSource       =
_bindingSource;
```

```
dataGridViewProductAccountings.Columns["ProductAccId"].HeaderTex
t = "ID";

dataGridViewProductAccountings.Columns["SupplyId"].HeaderText   =
"ID поставки";

dataGridViewProductAccountings.Columns["EmployeeId"].HeaderText
= "ID сотрудника";

dataGridViewProductAccountings.Columns["StorageId"].HeaderText  =
"ID зоны хранения";

dataGridViewProductAccountings.Columns["AccountingDate"].HeaderT
ext = "Дата учета";

dataGridViewProductAccountings.Columns["Quantity"].HeaderText   =
"Количество";

dataGridViewProductAccountings.Columns["LastMovementDate"].Heade
rText = "Дата движения";
            }
            catch (Exception ex)
            {
            MessageBox.Show($"Ошибка  загрузки  учета  продукции:
{ex.Message}",          "Ошибка",          MessageBoxButtons.OK,
MessageBoxIcon.Error);
            }
        }


        private void LoadSupplies()
        {
            try
            {
                var supplies = _supplyRepository.GetAll();
                cmbSupply.DataSource = supplies;
                cmbSupply.DisplayMember = "SupplyId";
                cmbSupply.ValueMember = "SupplyId";
            }
```

```
            catch (Exception ex)
            {
                MessageBox.Show($"Ошибка загрузки поставок: {ex.Message}", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);
            }
        }

        private void LoadEmployees()
        {
            try
            {
                var employees = _employeeRepository.GetAll();
                cmbEmployee.DataSource = employees;
                cmbEmployee.DisplayMember = "FullName";
                cmbEmployee.ValueMember = "EmployeeId";
            }
            catch (Exception ex)
            {
                MessageBox.Show($"Ошибка загрузки сотрудников: {ex.Message}", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);
            }
        }

        private void LoadStorageZones()
        {
            try
            {
            var storageZones = _storageZoneRepository.GetAll();
                cmbStorageZone.DataSource = storageZones;
                cmbStorageZone.DisplayMember = "ZoneName";
                cmbStorageZone.ValueMember = "StorageId";
            }
            catch (Exception ex)
```

```
            {
                MessageBox.Show($"Ошибка загрузки зон хранения: {ex.Message}", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);
            }
        }

        private void dataGridViewProductAccountings_SelectionChanged(object sender, EventArgs e)
        {
         if (dataGridViewProductAccountings.SelectedRows.Count > 0)
            {
                var selectedRow = dataGridViewProductAccountings.SelectedRows[0];
                _selectedProductAccounting = selectedRow.DataBoundItem as ProductAccounting;
                if (_selectedProductAccounting != null)
                {
                    cmbSupply.SelectedValue = _selectedProductAccounting.SupplyId;
                    cmbEmployee.SelectedValue = _selectedProductAccounting.EmployeeId;
                    cmbStorageZone.SelectedValue = _selectedProductAccounting.StorageId;
                    dtpAccountingDate.Value = _selectedProductAccounting.AccountingDate;
                    txtQuantity.Text = _selectedProductAccounting.Quantity.ToString();
                    if (_selectedProductAccounting.LastMovementDate.HasValue)
                    {
                        dtpLastMovementDate.Value = _selectedProductAccounting.LastMovementDate.Value;
                        chkLastMovementDate.Checked = true;
                    }
                    else
```

```
                {
                    chkLastMovementDate.Checked = false;
                }
            }
        }
    }

    private  void  chkLastMovementDate_CheckedChanged(object
sender, EventArgs e)
    {
        dtpLastMovementDate.Enabled                          =
chkLastMovementDate.Checked;
    }

    private void btnAdd_Click(object sender, EventArgs e)
    {
        try
        {
            if (!ValidateInput()) return;

            var productAccounting = new ProductAccounting
            {
                SupplyId = (int)cmbSupply.SelectedValue,
                EmployeeId = (int)cmbEmployee.SelectedValue,
              StorageId = (int)cmbStorageZone.SelectedValue,
                AccountingDate = dtpAccountingDate.Value,
                Quantity = int.Parse(txtQuantity.Text),
             LastMovementDate                                =
chkLastMovementDate.Checked ? dtpLastMovementDate.Value : null
            };

        _productAccountingRepository.Add(productAccounting);
            LoadProductAccountings();
            ClearInputs();
```

```
            }
            catch (Exception ex)
            {
            MessageBox.Show($"Ошибка добавления учета продукции: {ex.Message}",          "Ошибка",          MessageBoxButtons.OK, MessageBoxIcon.Error);
            }
        }

        private void btnUpdate_Click(object sender, EventArgs e)
        {
            try
            {
                if (_selectedProductAccounting == null)
                {
                MessageBox.Show("Выберите запись учета продукции для    обновления.",    "Предупреждение",    MessageBoxButtons.OK, MessageBoxIcon.Warning);
                    return;
                }

                if (!ValidateInput()) return;

                _selectedProductAccounting.SupplyId             = (int)cmbSupply.SelectedValue;
                _selectedProductAccounting.EmployeeId           = (int)cmbEmployee.SelectedValue;
                _selectedProductAccounting.StorageId            = (int)cmbStorageZone.SelectedValue;
                _selectedProductAccounting.AccountingDate       = dtpAccountingDate.Value;
                _selectedProductAccounting.Quantity             = int.Parse(txtQuantity.Text);
                _selectedProductAccounting.LastMovementDate     = chkLastMovementDate.Checked ? dtpLastMovementDate.Value : null;
```

```
_productAccountingRepository.Update(_selectedProductAccounting);
                LoadProductAccountings();
                ClearInputs();
            }
            catch (Exception ex)
            {
            MessageBox.Show($"Ошибка обновления учета продукции: {ex.Message}",          "Ошибка",          MessageBoxButtons.OK, MessageBoxIcon.Error);
            }
        }

        private void btnDelete_Click(object sender, EventArgs e)
        {
            try
            {
                if (_selectedProductAccounting == null)
                {
               MessageBox.Show("Выберите запись учета продукции для    удаления.",    "Предупреждение",    MessageBoxButtons.OK, MessageBoxIcon.Warning);
                    return;
                }

            if (MessageBox.Show($"Вы уверены, что хотите удалить запись учета с ID '{_selectedProductAccounting.ProductAccId}'?", "Подтверждение",                          MessageBoxButtons.YesNo, MessageBoxIcon.Question) == DialogResult.Yes)
                {

_productAccountingRepository.Delete(_selectedProductAccounting.ProductAccId);
                    LoadProductAccountings();
                    ClearInputs();
                }
            }
```

```
            catch (Exception ex)
            {
            MessageBox.Show($"Ошибка удаления учета продукции: {ex.Message}", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);
            }
        }

        private bool ValidateInput()
        {
            if (cmbSupply.SelectedValue == null)
            {
                MessageBox.Show("Выберите поставку.", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Warning);
                return false;
            }

            if (cmbEmployee.SelectedValue == null)
            {
            MessageBox.Show("Выберите сотрудника.", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Warning);
                return false;
            }

            if (cmbStorageZone.SelectedValue == null)
            {
            MessageBox.Show("Выберите зону хранения.", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Warning);
                return false;
            }

            if (string.IsNullOrWhiteSpace(txtQuantity.Text) || !int.TryParse(txtQuantity.Text, out int quantity) || quantity <= 0)
            {
```

```
                MessageBox.Show("Введите корректное количество
(положительное число).", "Ошибка", MessageBoxButtons.OK,
MessageBoxIcon.Warning);
                return false;
            }

            return true;
        }

        private void ClearInputs()
        {
            cmbSupply.SelectedIndex = -1;
            cmbEmployee.SelectedIndex = -1;
            cmbStorageZone.SelectedIndex = -1;
            dtpAccountingDate.Value = DateTime.Now;
            txtQuantity.Clear();
            dtpLastMovementDate.Value = DateTime.Now;
            chkLastMovementDate.Checked = false;
            _selectedProductAccounting = null;
        }
    }
}
```

Листинг файла ProductForm.cs:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.Xml.Linq;
```

```
using Application;
using Domain.Interfaces;
using Domain.Models;

namespace UI
{
    public partial class ProductForm : Form
    {
        private readonly IProductRepository _productRepository;
        private Product _selectedProduct;
        private BindingSource _bindingSource;
        private BindingNavigator _bindingNavigator;

        public ProductForm()
        {
            InitializeComponent();
            _productRepository                                    =
Application.AppContext.Instance.ProductRepository;
            _bindingSource = new BindingSource();

            _bindingNavigator                =                 new
BindingNavigator(_bindingSource);
            _bindingNavigator.Dock = DockStyle.Top;
            this.Controls.Add(_bindingNavigator);

         _bindingNavigator.MoveFirstItem.ToolTipText  =  "Перейти
к первой записи";
            _bindingNavigator.MovePreviousItem.ToolTipText      =
"Перейти к предыдущей записи";
         _bindingNavigator.MoveNextItem.ToolTipText   =   "Перейти
к следующей записи";
         _bindingNavigator.MoveLastItem.ToolTipText   =   "Перейти
к последней записи";
            _bindingNavigator.AddNewItem.ToolTipText  =  "Добавить
новую запись";
```

```
            _bindingNavigator.DeleteItem.ToolTipText = "Удалить
текущую запись";
          _bindingNavigator.PositionItem.ToolTipText = "Текущая
позиция";
            _bindingNavigator.CountItem.ToolTipText = "Общее
количество записей";
        }

       private void ProductForm_Load(object sender, EventArgs e)
        {
            LoadProducts();
        }

        private void LoadProducts()
        {
            try
            {
                var products = _productRepository.GetAll();
                _bindingSource.DataSource = products;
            dataGridViewProducts.DataSource = _bindingSource;

            dataGridViewProducts.Columns["ProductId"].HeaderText
= "ID";
                dataGridViewProducts.Columns["Name"].HeaderText
= "Название";

dataGridViewProducts.Columns["ExpiryDate"].HeaderText = "Срок
годности";

dataGridViewProducts.Columns["ProductType"].HeaderText = "Тип
продукта";
            dataGridViewProducts.Columns["IsActive"].HeaderText
= "Активен";
                dataGridViewProducts.Columns["Photo"].Visible =
false;
            }
            catch (Exception ex)
```

```
            {
                MessageBox.Show($"Ошибка загрузки продуктов: {ex.Message}", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);
            }
        }

      private void dataGridViewProducts_SelectionChanged(object sender, EventArgs e)
        {
            if (dataGridViewProducts.SelectedRows.Count > 0)
            {
                var selectedRow = dataGridViewProducts.SelectedRows[0];
                _selectedProduct = selectedRow.DataBoundItem as Product;
                if (_selectedProduct != null)
                {
                    txtName.Text = _selectedProduct.Name;
                dtpExpiryDate.Value = _selectedProduct.ExpiryDate ?? DateTime.Now;
                    dtpExpiryDate.Checked = _selectedProduct.ExpiryDate.HasValue;
                    cmbProductType.SelectedItem = _selectedProduct.ProductType.ToString();
                chkIsActive.Checked = _selectedProduct.IsActive;
                    if (_selectedProduct.Photo != null)
                    {
                        using (var ms = new System.IO.MemoryStream(_selectedProduct.Photo))
                        {
                            picPhoto.Image = System.Drawing.Image.FromStream(ms);
                        }
                    }
                    else
                    {
```

```
                    picPhoto.Image = null;
                }
            }
        }
    }

    private void btnUploadPhoto_Click(object sender, EventArgs
e)
    {
        using (var openFileDialog = new OpenFileDialog())
        {
            openFileDialog.Filter          =          "Image
Files|*.jpg;*.jpeg;*.png;*.bmp";
        if (openFileDialog.ShowDialog() == DialogResult.OK)
            {
                picPhoto.Image                                =
System.Drawing.Image.FromFile(openFileDialog.FileName);
            }
        }
    }

    private void btnAdd_Click(object sender, EventArgs e)
    {
        try
        {
            if (!ValidateInput()) return;

            var product = new Product
            {
                Name = txtName.Text,
                ExpiryDate    =    dtpExpiryDate.Checked    ?
dtpExpiryDate.Value : null,
                ProductType                                   =
(ProductType)Enum.Parse(typeof(ProductType),
cmbProductType.SelectedItem.ToString()),
```

```
                    IsActive = chkIsActive.Checked,
                    Photo = picPhoto.Image != null ? ImageToByteArray(picPhoto.Image) : null
                };

                _productRepository.Add(product);
                LoadProducts();
                ClearInputs();
            }
            catch (Exception ex)
            {
                MessageBox.Show($"Ошибка добавления продукта: {ex.Message}", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);
            }
        }

        private void btnUpdate_Click(object sender, EventArgs e)
        {
            try
            {
                if (_selectedProduct == null)
                {
                    MessageBox.Show("Выберите продукт для обновления.", "Предупреждение", MessageBoxButtons.OK, MessageBoxIcon.Warning);
                    return;
                }
                if (!ValidateInput()) return;

                _selectedProduct.Name = txtName.Text;
            _selectedProduct.ExpiryDate = dtpExpiryDate.Checked ? dtpExpiryDate.Value : null;
                _selectedProduct.ProductType = (ProductType)Enum.Parse(typeof(ProductType), cmbProductType.SelectedItem.ToString());
```

```
                _selectedProduct.IsActive = chkIsActive.Checked;
                _selectedProduct.Photo   =   picPhoto.Image   !=
null ? ImageToByteArray(picPhoto.Image) : null;

                _productRepository.Update(_selectedProduct);
                LoadProducts();
                ClearInputs();
            }
            catch (Exception ex)
            {
                MessageBox.Show($"Ошибка   обновления   продукта:
{ex.Message}",          "Ошибка",          MessageBoxButtons.OK,
MessageBoxIcon.Error);
            }
        }

        private void btnDelete_Click(object sender, EventArgs e)
        {
            try
            {
                if (_selectedProduct == null)
                {
               MessageBox.Show("Выберите продукт для удаления.",
"Предупреждение", MessageBoxButtons.OK, MessageBoxIcon.Warning);
                    return;
                }

            if (MessageBox.Show($"Вы уверены, что хотите удалить
продукт       '{_selectedProduct.Name}'?",       "Подтверждение",
MessageBoxButtons.YesNo,       MessageBoxIcon.Question)       ==
DialogResult.Yes)
                {

_productRepository.Delete(_selectedProduct.ProductId);
                    LoadProducts();
                    ClearInputs();
```

```
            }
        }
        catch (Exception ex)
        {
            MessageBox.Show($"Ошибка удаления продукта: {ex.Message}", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);
        }
    }

    private bool ValidateInput()
    {
        if (string.IsNullOrWhiteSpace(txtName.Text))
        {
            MessageBox.Show("Введите название продукта.", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Warning);
            return false;
        }

        if (cmbProductType.SelectedItem == null)
        {
            MessageBox.Show("Выберите тип продукта.", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Warning);
            return false;
        }

        return true;
    }

    private void ClearInputs()
    {
        txtName.Clear();
        dtpExpiryDate.Checked = false;
        dtpExpiryDate.Value = DateTime.Now;
        cmbProductType.SelectedItem = null;
```

```
            chkIsActive.Checked = true;
            picPhoto.Image = null;
            _selectedProduct = null;
        }

      private byte[] ImageToByteArray(System.Drawing.Image image)
        {
            using (var ms = new System.IO.MemoryStream())
            {
                image.Save(ms, image.RawFormat);
                return ms.ToArray();
            }
        }
    }
}
```

Листинг файла StorageZoneForm.cs:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using Domain.Interfaces;
using Domain.Models;

namespace UI
{
    public partial class StorageZoneForm : Form
    {
```

```
        private          readonly          IStorageZoneRepository
_storageZoneRepository;
      private readonly IWarehouseRepository _warehouseRepository;
        private StorageZone _selectedStorageZone;
        private BindingSource _bindingSource;
        private BindingNavigator _bindingNavigator;


        public StorageZoneForm()
        {
            InitializeComponent();
            _storageZoneRepository                                    =
Application.AppContext.Instance.StorageZoneRepository;
            _warehouseRepository                                      =
Application.AppContext.Instance.WarehouseRepository;
            _bindingSource = new BindingSource();


            _bindingNavigator                =                 new
BindingNavigator(_bindingSource);
            _bindingNavigator.Dock = DockStyle.Top;
            this.Controls.Add(_bindingNavigator);


         _bindingNavigator.MoveFirstItem.ToolTipText  =  "Перейти
к первой записи";
            _bindingNavigator.MovePreviousItem.ToolTipText      =
"Перейти к предыдущей записи";
         _bindingNavigator.MoveNextItem.ToolTipText   =   "Перейти
к следующей записи";
         _bindingNavigator.MoveLastItem.ToolTipText   =   "Перейти
к последней записи";
            _bindingNavigator.AddNewItem.ToolTipText  =  "Добавить
новую запись";
            _bindingNavigator.DeleteItem.ToolTipText  =  "Удалить
текущую запись";
         _bindingNavigator.PositionItem.ToolTipText   =   "Текущая
позиция";
            _bindingNavigator.CountItem.ToolTipText    =    "Общее
количество записей";
        }
```

```
      private void StorageZoneForm_Load(object sender, EventArgs
e)
        {
            LoadStorageZones();
            LoadWarehouses();
            LoadZoneTypes();
        }

        private void LoadStorageZones()
        {
            try
            {
            var storageZones = _storageZoneRepository.GetAll();
                _bindingSource.DataSource = storageZones;
            dataGridViewStorageZones.DataSource = _bindingSource;


dataGridViewStorageZones.Columns["StorageId"].HeaderText = "ID";

dataGridViewStorageZones.Columns["WarehouseId"].HeaderText = "ID
склада";

dataGridViewStorageZones.Columns["Capacity"].HeaderText =
"Вместимость";

dataGridViewStorageZones.Columns["ZoneType"].HeaderText = "Тип
зоны";

dataGridViewStorageZones.Columns["ZoneName"].HeaderText =
"Название зоны";
            }
            catch (Exception ex)
            {
                MessageBox.Show($"Ошибка загрузки зон хранения:
{ex.Message}", "Ошибка", MessageBoxButtons.OK,
MessageBoxIcon.Error);
            }
```

```
        }

        private void LoadWarehouses()
        {
            try
            {
                var warehouses = _warehouseRepository.GetAll();
                cmbWarehouse.DataSource = warehouses;
                cmbWarehouse.DisplayMember = "Name";
                cmbWarehouse.ValueMember = "WarehouseId";
            }
            catch (Exception ex)
            {
                MessageBox.Show($"Ошибка загрузки складов: {ex.Message}", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);
            }
        }

        private void LoadZoneTypes()
        {
            try
            {
            var zoneTypes = Enum.GetValues(typeof(ZoneType));
                cmbZoneType.DataSource = zoneTypes;
            }
            catch (Exception ex)
            {
                MessageBox.Show($"Ошибка загрузки типов зон: {ex.Message}", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);
            }
        }
```

```
        private                                                 void
dataGridViewStorageZones_SelectionChanged(object          sender,
EventArgs e)
        {
            if (dataGridViewStorageZones.SelectedRows.Count > 0)
            {
                var                 selectedRow                 =
dataGridViewStorageZones.SelectedRows[0];
                _selectedStorageZone  =  selectedRow.DataBoundItem
as StorageZone;
                if (_selectedStorageZone != null)
                {
                    cmbWarehouse.SelectedValue                  =
_selectedStorageZone.WarehouseId;
                    txtCapacity.Text                            =
_selectedStorageZone.Capacity.ToString();
                    cmbZoneType.SelectedItem                    =
_selectedStorageZone.ZoneType;
               txtZoneName.Text = _selectedStorageZone.ZoneName;
                }
            }
        }

        private void btnAdd_Click(object sender, EventArgs e)
        {
            try
            {
                if (!ValidateInput()) return;

                var storageZone = new StorageZone
                {
               WarehouseId = (int)cmbWarehouse.SelectedValue,
                    Capacity = int.Parse(txtCapacity.Text),
               ZoneType = (ZoneType)cmbZoneType.SelectedItem,
                    ZoneName = txtZoneName.Text
                };
```

```
                _storageZoneRepository.Add(storageZone);
                LoadStorageZones();
                ClearInputs();
            }
            catch (Exception ex)
            {
            MessageBox.Show($"Ошибка добавления зоны хранения: {ex.Message}", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);
            }
        }

        private void btnUpdate_Click(object sender, EventArgs e)
        {
            try
            {
                if (_selectedStorageZone == null)
                {
                    MessageBox.Show("Выберите зону хранения для обновления.", "Предупреждение", MessageBoxButtons.OK, MessageBoxIcon.Warning);
                    return;
                }

                if (!ValidateInput()) return;

                _selectedStorageZone.WarehouseId = (int)cmbWarehouse.SelectedValue;
                _selectedStorageZone.Capacity = int.Parse(txtCapacity.Text);
                _selectedStorageZone.ZoneType = (ZoneType)cmbZoneType.SelectedItem;
            _selectedStorageZone.ZoneName = txtZoneName.Text;

            _storageZoneRepository.Update(_selectedStorageZone);
```

```
                LoadStorageZones();
                ClearInputs();
            }
            catch (Exception ex)
            {
            MessageBox.Show($"Ошибка обновления зоны хранения: {ex.Message}", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);
            }
        }

        private void btnDelete_Click(object sender, EventArgs e)
        {
            try
            {
                if (_selectedStorageZone == null)
                {
                    MessageBox.Show("Выберите зону хранения для удаления.", "Предупреждение", MessageBoxButtons.OK, MessageBoxIcon.Warning);
                    return;
                }

            if (MessageBox.Show($"Вы уверены, что хотите удалить зону хранения '{_selectedStorageZone.ZoneName}'?", "Подтверждение", MessageBoxButtons.YesNo, MessageBoxIcon.Question) == DialogResult.Yes)
                {

_storageZoneRepository.Delete(_selectedStorageZone.StorageId);
                    LoadStorageZones();
                    ClearInputs();
                }
            }
            catch (Exception ex)
            {
```

```
                MessageBox.Show($"Ошибка удаления зоны хранения:
{ex.Message}",          "Ошибка",          MessageBoxButtons.OK,
MessageBoxIcon.Error);
            }
        }

        private bool ValidateInput()
        {
            if (cmbWarehouse.SelectedValue == null)
            {
                MessageBox.Show("Выберите   склад.",   "Ошибка",
MessageBoxButtons.OK, MessageBoxIcon.Warning);
                return false;
            }

            if       (string.IsNullOrWhiteSpace(txtCapacity.Text)
|| !int.TryParse(txtCapacity.Text, out int capacity) || capacity
<= 0)
            {
                MessageBox.Show("Введите   корректную   вместимость
(положительное    число).",    "Ошибка",    MessageBoxButtons.OK,
MessageBoxIcon.Warning);
                return false;
            }

            if (cmbZoneType.SelectedItem == null)
            {
                MessageBox.Show("Выберите тип зоны.", "Ошибка",
MessageBoxButtons.OK, MessageBoxIcon.Warning);
                return false;
            }

            if (string.IsNullOrWhiteSpace(txtZoneName.Text))
            {
            MessageBox.Show("Введите название зоны.", "Ошибка",
MessageBoxButtons.OK, MessageBoxIcon.Warning);
                return false;
```

```
            }

            return true;
        }

        private void ClearInputs()
        {
            cmbWarehouse.SelectedIndex = -1;
            txtCapacity.Clear();
            cmbZoneType.SelectedIndex = -1;
            txtZoneName.Clear();
            _selectedStorageZone = null;
        }
    }
}
```

Листинг файла SupplierForm.cs:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using Domain.Interfaces;
using Domain.Models;

namespace UI
{
    public partial class SupplierForm : Form
    {
```

```
      private readonly ISupplierRepository _supplierRepository;
        private Supplier _selectedSupplier;
        private BindingSource _bindingSource;
        private BindingNavigator _bindingNavigator;

        public SupplierForm()
        {
            InitializeComponent();
            _supplierRepository                                    =
Application.AppContext.Instance.SupplierRepository;
            _bindingSource = new BindingSource();

            _bindingNavigator                =                 new
BindingNavigator(_bindingSource);
            _bindingNavigator.Dock = DockStyle.Top;
            this.Controls.Add(_bindingNavigator);

         _bindingNavigator.MoveFirstItem.ToolTipText  =  "Перейти
к первой записи";
            _bindingNavigator.MovePreviousItem.ToolTipText      =
"Перейти к предыдущей записи";
         _bindingNavigator.MoveNextItem.ToolTipText   =   "Перейти
к следующей записи";
         _bindingNavigator.MoveLastItem.ToolTipText   =   "Перейти
к последней записи";
            _bindingNavigator.AddNewItem.ToolTipText  =  "Добавить
новую запись";
            _bindingNavigator.DeleteItem.ToolTipText  =  "Удалить
текущую запись";
         _bindingNavigator.PositionItem.ToolTipText   =   "Текущая
позиция";
            _bindingNavigator.CountItem.ToolTipText    =    "Общее
количество записей";
        }

        private  void  SupplierForm_Load(object  sender,  EventArgs
e)
```

```
        {
            LoadSuppliers();
        }

        private void LoadSuppliers()
        {
            try
            {
                var suppliers = _supplierRepository.GetAll();
                _bindingSource.DataSource = suppliers;
            dataGridViewSuppliers.DataSource = _bindingSource;


dataGridViewSuppliers.Columns["SupplierId"].HeaderText = "ID";

dataGridViewSuppliers.Columns["CompanyName"].HeaderText         =
"Название компании";

dataGridViewSuppliers.Columns["ContactPerson"].HeaderText       =
"Контактное лицо";
            dataGridViewSuppliers.Columns["Phone"].HeaderText
= "Телефон";
            dataGridViewSuppliers.Columns["Address"].HeaderText
= "Адрес";
            }
            catch (Exception ex)
            {
                MessageBox.Show($"Ошибка  загрузки  поставщиков:
{ex.Message}",          "Ошибка",          MessageBoxButtons.OK,
MessageBoxIcon.Error);
            }
        }


      private void dataGridViewSuppliers_SelectionChanged(object
sender, EventArgs e)
        {
            if (dataGridViewSuppliers.SelectedRows.Count > 0)
```

```
            {
                var                 selectedRow                 =
dataGridViewSuppliers.SelectedRows[0];
                _selectedSupplier = selectedRow.DataBoundItem as
Supplier;
                if (_selectedSupplier != null)
                {
                    txtCompanyName.Text                         =
_selectedSupplier.CompanyName;
                    txtContactPerson.Text                       =
_selectedSupplier.ContactPerson;
                    txtPhone.Text = _selectedSupplier.Phone;
                    txtAddress.Text = _selectedSupplier.Address;
                }
            }
        }

        private void btnAdd_Click(object sender, EventArgs e)
        {
            try
            {
                if (!ValidateInput()) return;

                var supplier = new Supplier
                {
                    CompanyName = txtCompanyName.Text,
                    ContactPerson = txtContactPerson.Text,
                    Phone = txtPhone.Text,
                    Address = txtAddress.Text
                };

                _supplierRepository.Add(supplier);
                LoadSuppliers();
                ClearInputs();
            }
```

```
            catch (Exception ex)
            {
                MessageBox.Show($"Ошибка добавления поставщика: {ex.Message}", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);
            }
        }

        private void btnUpdate_Click(object sender, EventArgs e)
        {
            try
            {
                if (_selectedSupplier == null)
                {
                    MessageBox.Show("Выберите поставщика для обновления.", "Предупреждение", MessageBoxButtons.OK, MessageBoxIcon.Warning);
                    return;
                }

                if (!ValidateInput()) return;

            _selectedSupplier.CompanyName = txtCompanyName.Text;
                _selectedSupplier.ContactPerson = txtContactPerson.Text;
                _selectedSupplier.Phone = txtPhone.Text;
                _selectedSupplier.Address = txtAddress.Text;

                _supplierRepository.Update(_selectedSupplier);
                LoadSuppliers();
                ClearInputs();
            }
            catch (Exception ex)
            {
```

```
                MessageBox.Show($"Ошибка обновления поставщика: {ex.Message}", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);
            }
        }

        private void btnDelete_Click(object sender, EventArgs e)
        {
            try
            {
                if (_selectedSupplier == null)
                {
                    MessageBox.Show("Выберите поставщика для удаления.", "Предупреждение", MessageBoxButtons.OK, MessageBoxIcon.Warning);
                    return;
                }

            if (MessageBox.Show($"Вы уверены, что хотите удалить поставщика '{_selectedSupplier.CompanyName}'?", "Подтверждение", MessageBoxButtons.YesNo, MessageBoxIcon.Question) == DialogResult.Yes)
                {

_supplierRepository.Delete(_selectedSupplier.SupplierId);
                    LoadSuppliers();
                    ClearInputs();
                }
            }
            catch (Exception ex)
            {
                MessageBox.Show($"Ошибка удаления поставщика: {ex.Message}", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);
            }
        }
```

```
        private bool ValidateInput()
        {
            if (string.IsNullOrWhiteSpace(txtCompanyName.Text))
            {
                MessageBox.Show("Введите название компании.",
"Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Warning);
                return false;
            }

            return true;
        }

        private void ClearInputs()
        {
            txtCompanyName.Clear();
            txtContactPerson.Clear();
            txtPhone.Clear();
            txtAddress.Clear();
            _selectedSupplier = null;
        }
    }
}
```

Листинг файла SupplyForm.cs:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
```

```
using Domain.Interfaces;
using Domain.Models;

namespace UI
{
    public partial class SupplyForm : Form
    {
        private readonly ISupplyRepository _supplyRepository;
        private readonly IProductRepository _productRepository;
      private readonly ISupplierRepository _supplierRepository;
        private Supply _selectedSupply;
        private BindingSource _bindingSource;
        private BindingNavigator _bindingNavigator;

        public SupplyForm()
        {
            InitializeComponent();
            _supplyRepository                                    =
Application.AppContext.Instance.SupplyRepository;
            _productRepository                                   =
Application.AppContext.Instance.ProductRepository;
            _supplierRepository                                  =
Application.AppContext.Instance.SupplierRepository;
            _bindingSource = new BindingSource();

            _bindingNavigator                =                new
BindingNavigator(_bindingSource);
            _bindingNavigator.Dock = DockStyle.Top;
            this.Controls.Add(_bindingNavigator);

         _bindingNavigator.MoveFirstItem.ToolTipText  =  "Перейти
к первой записи";
            _bindingNavigator.MovePreviousItem.ToolTipText      =
"Перейти к предыдущей записи";
         _bindingNavigator.MoveNextItem.ToolTipText   =   "Перейти
к следующей записи";
```

```
         _bindingNavigator.MoveLastItem.ToolTipText = "Перейти
к последней записи";
            _bindingNavigator.AddNewItem.ToolTipText = "Добавить
новую запись";
            _bindingNavigator.DeleteItem.ToolTipText = "Удалить
текущую запись";
         _bindingNavigator.PositionItem.ToolTipText = "Текущая
позиция";
            _bindingNavigator.CountItem.ToolTipText = "Общее
количество записей";
        }

        private void SupplyForm_Load(object sender, EventArgs e)
        {
            LoadSupplies();
            LoadProducts();
            LoadSuppliers();
        }

        private void LoadSupplies()
        {
            try
            {
                var supplies = _supplyRepository.GetAll();
                _bindingSource.DataSource = supplies;
            dataGridViewSupplies.DataSource = _bindingSource;

            dataGridViewSupplies.Columns["SupplyId"].HeaderText
= "ID";
            dataGridViewSupplies.Columns["ProductId"].HeaderText
= "ID продукта";

dataGridViewSupplies.Columns["SupplierId"].HeaderText = "ID
поставщика";

dataGridViewSupplies.Columns["SupplyDate"].HeaderText = "Дата
поставки";
```

```
            dataGridViewSupplies.Columns["Quantity"].HeaderText
= "Количество";
            }
            catch (Exception ex)
            {
                MessageBox.Show($"Ошибка    загрузки    поставок:
{ex.Message}",          "Ошибка",          MessageBoxButtons.OK,
MessageBoxIcon.Error);
            }
        }

        private void LoadProducts()
        {
            try
            {
                var products = _productRepository.GetAll();
                cmbProduct.DataSource = products;
                cmbProduct.DisplayMember = "Name";
                cmbProduct.ValueMember = "ProductId";
            }
            catch (Exception ex)
            {
                MessageBox.Show($"Ошибка    загрузки    продуктов:
{ex.Message}",          "Ошибка",          MessageBoxButtons.OK,
MessageBoxIcon.Error);
            }
        }

        private void LoadSuppliers()
        {
            try
            {
                var suppliers = _supplierRepository.GetAll();
                cmbSupplier.DataSource = suppliers;
                cmbSupplier.DisplayMember = "CompanyName";
```

```
                cmbSupplier.ValueMember = "SupplierId";
            }
            catch (Exception ex)
            {
                MessageBox.Show($"Ошибка загрузки поставщиков: {ex.Message}", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);
            }
        }

      private void dataGridViewSupplies_SelectionChanged(object sender, EventArgs e)
        {
            if (dataGridViewSupplies.SelectedRows.Count > 0)
            {
                var selectedRow = dataGridViewSupplies.SelectedRows[0];
                _selectedSupply = selectedRow.DataBoundItem as Supply;
                if (_selectedSupply != null)
                {
                    cmbProduct.SelectedValue = _selectedSupply.ProductId;
                    cmbSupplier.SelectedValue = _selectedSupply.SupplierId;
               dtpSupplyDate.Value = _selectedSupply.SupplyDate;
                    txtQuantity.Text = _selectedSupply.Quantity.ToString();
                }
            }
        }

        private void btnAdd_Click(object sender, EventArgs e)
        {
            try
            {
```

```
                if (!ValidateInput()) return;

                var supply = new Supply
                {
                    ProductId = (int)cmbProduct.SelectedValue,
                    SupplierId = (int)cmbSupplier.SelectedValue,
                    SupplyDate = dtpSupplyDate.Value,
                    Quantity = int.Parse(txtQuantity.Text)
                };

                _supplyRepository.Add(supply);
                LoadSupplies();
                ClearInputs();
            }
            catch (Exception ex)
            {
                MessageBox.Show($"Ошибка добавления поставки: {ex.Message}", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);
            }
        }

        private void btnUpdate_Click(object sender, EventArgs e)
        {
            try
            {
                if (_selectedSupply == null)
                {
                    MessageBox.Show("Выберите поставку для обновления.", "Предупреждение", MessageBoxButtons.OK, MessageBoxIcon.Warning);
                    return;
                }

                if (!ValidateInput()) return;
```

```
                _selectedSupply.ProductId                    =
(int)cmbProduct.SelectedValue;
                _selectedSupply.SupplierId                   =
(int)cmbSupplier.SelectedValue;
            _selectedSupply.SupplyDate = dtpSupplyDate.Value;
                _selectedSupply.Quantity                     =
int.Parse(txtQuantity.Text);


                _supplyRepository.Update(_selectedSupply);
                LoadSupplies();
                ClearInputs();
            }
            catch (Exception ex)
            {
                MessageBox.Show($"Ошибка   обновления   поставки:
{ex.Message}",          "Ошибка",          MessageBoxButtons.OK,
MessageBoxIcon.Error);
            }
        }


        private void btnDelete_Click(object sender, EventArgs e)
        {
            try
            {
                if (_selectedSupply == null)
                {
               MessageBox.Show("Выберите поставку для удаления.",
"Предупреждение", MessageBoxButtons.OK, MessageBoxIcon.Warning);
                    return;
                }


            if (MessageBox.Show($"Вы уверены, что хотите удалить
поставку  с  ID  '{_selectedSupply.SupplyId}'?",  "Подтверждение",
MessageBoxButtons.YesNo,       MessageBoxIcon.Question)       ==
DialogResult.Yes)
                {
```

```
_supplyRepository.Delete(_selectedSupply.SupplyId);
                    LoadSupplies();
                    ClearInputs();
                }
            }
            catch (Exception ex)
            {
                MessageBox.Show($"Ошибка удаления поставки: {ex.Message}", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);
            }
        }

        private bool ValidateInput()
        {
            if (cmbProduct.SelectedValue == null)
            {
                MessageBox.Show("Выберите продукт.", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Warning);
                return false;
            }

            if (cmbSupplier.SelectedValue == null)
            {
            MessageBox.Show("Выберите поставщика.", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Warning);
                return false;
            }

            if (string.IsNullOrWhiteSpace(txtQuantity.Text) || !int.TryParse(txtQuantity.Text, out int quantity) || quantity <= 0)
            {
                MessageBox.Show("Введите корректное количество (положительное число).", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Warning);
```

```
                return false;
            }

            return true;
        }

        private void ClearInputs()
        {
            cmbProduct.SelectedIndex = -1;
            cmbSupplier.SelectedIndex = -1;
            dtpSupplyDate.Value = DateTime.Now;
            txtQuantity.Clear();
            _selectedSupply = null;
        }
    }
}
```

Листинг файла WarehouseForm.cs:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using Domain.Interfaces;
using System.Xml.Linq;
using Domain.Models;

namespace UI
{
```

```
    public partial class WarehouseForm : Form
    {
      private readonly IWarehouseRepository _warehouseRepository;
        private Warehouse _selectedWarehouse;
        private BindingSource _bindingSource;
        private BindingNavigator _bindingNavigator;

        public WarehouseForm()
        {
            InitializeComponent();
            _warehouseRepository                                    =
Application.AppContext.Instance.WarehouseRepository;
            _bindingSource = new BindingSource();

            _bindingNavigator                =                 new
BindingNavigator(_bindingSource);
            _bindingNavigator.Dock = DockStyle.Top;
            this.Controls.Add(_bindingNavigator);

         _bindingNavigator.MoveFirstItem.ToolTipText  =  "Перейти
к первой записи";
            _bindingNavigator.MovePreviousItem.ToolTipText      =
"Перейти к предыдущей записи";
         _bindingNavigator.MoveNextItem.ToolTipText   =   "Перейти
к следующей записи";
         _bindingNavigator.MoveLastItem.ToolTipText   =   "Перейти
к последней записи";
            _bindingNavigator.AddNewItem.ToolTipText  =  "Добавить
новую запись";
            _bindingNavigator.DeleteItem.ToolTipText  =  "Удалить
текущую запись";
         _bindingNavigator.PositionItem.ToolTipText   =   "Текущая
позиция";
            _bindingNavigator.CountItem.ToolTipText    =    "Общее
количество записей";
        }
```

```
        private void WarehouseForm_Load(object sender, EventArgs
e)
        {
            LoadWarehouses();
        }

        private void LoadWarehouses()
        {
            try
            {
                var warehouses = _warehouseRepository.GetAll();
                _bindingSource.DataSource = warehouses;
            dataGridViewWarehouses.DataSource = _bindingSource;


dataGridViewWarehouses.Columns["WarehouseId"].HeaderText = "ID";
            dataGridViewWarehouses.Columns["Name"].HeaderText
= "Название";
            dataGridViewWarehouses.Columns["Address"].HeaderText
= "Адрес";
            }
            catch (Exception ex)
            {
                MessageBox.Show($"Ошибка загрузки складов:
{ex.Message}",          "Ошибка",          MessageBoxButtons.OK,
MessageBoxIcon.Error);
            }
        }


      private void dataGridViewWarehouses_SelectionChanged(object
sender, EventArgs e)
        {
            if (dataGridViewWarehouses.SelectedRows.Count > 0)
            {
                var                selectedRow                =
dataGridViewWarehouses.SelectedRows[0];
```

```
            _selectedWarehouse = selectedRow.DataBoundItem as
Warehouse;
                if (_selectedWarehouse != null)
                {
                    txtName.Text = _selectedWarehouse.Name;
               txtAddress.Text = _selectedWarehouse.Address;
                }
            }
        }

        private void btnAdd_Click(object sender, EventArgs e)
        {
            try
            {
                if (!ValidateInput()) return;

                var warehouse = new Warehouse
                {
                    Name = txtName.Text,
                    Address = txtAddress.Text
                };

                _warehouseRepository.Add(warehouse);
                LoadWarehouses();
                ClearInputs();
            }
            catch (Exception ex)
            {
                MessageBox.Show($"Ошибка добавления склада:
{ex.Message}", "Ошибка", MessageBoxButtons.OK,
MessageBoxIcon.Error);
            }
        }
```

```
        private void btnUpdate_Click(object sender, EventArgs e)
        {
            try
            {
                if (_selectedWarehouse == null)
                {
                   MessageBox.Show("Выберите склад для обновления.",
"Предупреждение", MessageBoxButtons.OK, MessageBoxIcon.Warning);
                    return;
                }
                if (!ValidateInput()) return;

                _selectedWarehouse.Name = txtName.Text;
                _selectedWarehouse.Address = txtAddress.Text;

                _warehouseRepository.Update(_selectedWarehouse);
                LoadWarehouses();
                ClearInputs();
            }
            catch (Exception ex)
            {
                MessageBox.Show($"Ошибка    обновления    склада:
{ex.Message}",          "Ошибка",          MessageBoxButtons.OK,
MessageBoxIcon.Error);
            }
        }

        private void btnDelete_Click(object sender, EventArgs e)
        {
            try
            {
                if (_selectedWarehouse == null)
                {
                   MessageBox.Show("Выберите склад для удаления.",
"Предупреждение", MessageBoxButtons.OK, MessageBoxIcon.Warning);
```

```
                return;
            }

            if (MessageBox.Show($"Вы уверены, что хотите удалить склад '{_selectedWarehouse.Name}'?", "Подтверждение", MessageBoxButtons.YesNo, MessageBoxIcon.Question) == DialogResult.Yes)
            {
                _warehouseRepository.Delete(_selectedWarehouse.WarehouseId);
                LoadWarehouses();
                ClearInputs();
            }
        }
        catch (Exception ex)
        {
            MessageBox.Show($"Ошибка удаления склада: {ex.Message}", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);
        }
    }

    private bool ValidateInput()
    {
        if (string.IsNullOrWhiteSpace(txtName.Text))
        {
            MessageBox.Show("Введите название склада.", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Warning);
            return false;
        }

        return true;
    }

    private void ClearInputs()
    {
```

```
            txtName.Clear();
            txtAddress.Clear();
            _selectedWarehouse = null;
        }
    }
}
```